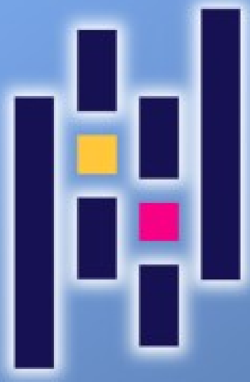

எளிய தமிழில் pandas

து.நித்யா

[illegible] Pandas

[illegible]

nithyadurai87@gmail.com

[illegible] :

[illegible]

kaniyam.com

எளிய தமிழில் Pandas Kaniyam.com

அட்டைப்படம் - லெனின் குருசாமி guruleninn@gmail.com

மின்னூலாக்கம்: த. சீனிவாசன் tshrinivasan@gmail.com

உரிமை:

Creative Commons Attribution - ShareAlike 4.0 International License.

எளிய தமிழில் Pandas Kaniyam.com

Pandas என்பது Python மொழி மூலம் வெள்ளமெனப் பெருகி வரும் தகவல்களை எளிதில் கையாள உதவுகிறது.

இந்த நூலைப் படிக்க, பைத்தான் மொழியின் அறிமுகம் அவசியம்.

பல்வேறு வகைகளில், வடிவங்களில் தகவல் இருப்பதால், அவற்றில் இருந்து தேவையான விவரங்களைப் பெறுவது கடினம். ஆனால் ஓஓஓ மூலம், தகவல்களை எளிதில் உருமாற்றி, அவற்றின் பின் உள்ள விவரங்களைப் பெற்று, முக்கிய முடிவுகளை எடுக்கப் பயன்படுத்தலாம்.

சென்னையில் நடைபெற்று வரும் மூன்றாவது 'தமிழ் கட்டற்ற மென்பொருள் மாநாடு 2023' ல் இந்த மின்னூலை வெளியிடுவதில் பெருமகிழ்ச்சி.

தமிழில் கட்டற்ற மென்பொருட்கள் பற்றிய தகவல்களை “கணியம்" மின் இதழ், 2012 முதல் வெளியிட்டு வருகிறது.இதில் வெளியான Pandas பற்றிய கட்டுரைககளை இணைத்து ஒரு முழு புத்தகமாக வெளியிடுவதில் பெரு மகிழ்ச்சி கொள்கிறோம்.

உங்கள் கருத்துகளையும், பிழை திருத்தங்களையும் editor@kaniyam.com க்கு மின்னஞ்சல் அனுப்பலாம்.

https://kaniyam.com/learn-pandas-in-tamil-ebook என்ற முகவரியில் இருந்து இந்த நூலை பதிவிறக்கம் செய்யலாம். உங்கள் கருத்துகளையும் இங்கே பகிரலாம்.

படித்து பயன் பெறவும், பிறருடன் பகிர்ந்து மகிழவும் வேண்டுகிறோம்.

கணியம் இதழை தொடர்ந்து வளர்க்கும் அனைத்து அன்பர்களுக்கும் எமது நன்றிகள்.

த.சீனிவாசன்

tshrinivasan@gmail.com

ஆசிரியர்
கணியம்

editor@kaniyam.com

உரிமை

இந்த நூல் கிரியேடிவ் காமன்ஸ் என்ற உரிமையில் வெளியிடப்படுகிறது . இதன் மூலம், நீங்கள்

- யாருடனும் பகிர்ந்து கொள்ளலாம்.
- திருத்தி எழுதி வெளியிடலாம்.
- வணிக ரீதியிலும்யன்படுத்தலாம்.

ஆனால், மூலப் புத்தகம் , ஆசிரியர் மற்றும் www.kaniyam.com பற்றிய விவரங்களை சேர்த்து தர வேண்டும். இதே உரிமைகளை யாவருக்கும் தர வேண்டும். கிரியேடிவ் காமன்ஸ் என்ற உரிமையில் வெளியிட வேண்டும்.

நூல் மூலம் :

http://static.kaniyam.com/ebooks/Learn-Pandas-in-Tamil.odt

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

எளிய தமிழில் Pandas Kaniyam.com

ஆசிரியர் உரை

பாண்டாஸ் பற்றிய என்னுடைய ஆசிரியர் உரையைத் துவங்குவதற்கு முன் ஒரு சின்ன பிளாஷ்பேக். அப்போதுதான் உங்களால் எதற்காக நான் இவ்வாறு எழுதியுள்ளேன் என்பதைப் புரிந்து கொள்ள முடியும்.

நான் கல்லூரி படிக்கும் போது Data Structures என்றொரு பாடம் உண்டு. அப்பாடவேளையின்போது Data)(என போர்டில் எழுதிப்போட்டு அனைவரையும் சிரிக்க வைத்துக் கொண்டிருந்தேன். ஆசிரியர் வருவதற்குள் அழித்து விடலாம் என்றே நினைத்தேன். ஆனால் அதற்குள் வந்துவிட்டார். போர்டில் Data)(என எழுதியிருப்பதைப் பார்த்து பொம்பள புள்ளங்க மாதிரியா நடந்துக்குறீங்க (அது மகளிர் கல்லூரி) எனத் திட்டிவிட்டு பாடம் எடுக்கத் தொடங்கினார்.

அன்று முதல் இன்று வரை Data Structure என்றாலே ஒரு நல்ல உடற்கட்டு வடிவம்தான் என் நினைவுக்கு வரும். இது வயதுக் கோளாறா அல்லது சினிமாவின் தாக்கமா என்பது தெரியவில்லை. வயதுக் கோளாறாக இருந்தால் அது இந்நேரம் சரியாகியிருக்கும். ஏனெனில் கோளாறு பிடித்த வயதை நான் எப்போதோ தாண்டிவிட்டேன். ஆனால் இப்போதும் கூட data structure அல்லது structure of algorithm என்பது போன்ற வார்த்தைகளைக் கேட்டாலோ அல்லது ஸ்ட்ரக்சர் எனும் வார்த்தையை வேறு எந்த ஒரு வார்த்தையுடன் தொடர்புபடுத்திக் கேட்டாலோ உடனே என் நினைவுக்கு வருவது ஒரு நல்ல உடற்கட்டு வடிவம் தான். ஆகவே தான் பல்வேறு டேட்டா ஸ்ட்ரக்சரைக் கையாள்வதற்கு உதவும் பாண்டாசைப் புரிந்து கொள்வதிலும் நான் இதே வழிமுறையைக் கையாண்டுள்ளேன். அதையே இங்கு ஆசிரியர் உரையாகவும் எழுதியுள்ளேன்.

மாடலிங், பளு தூக்குதல், டான்ஸ், அதெலடிக்ஸ் போன்ற பல்வேறு துறைகளில் ஜொலிக்க விரும்பும் பெண்கள், அத்துறைகளில் நேரடியாக களம் இறங்குவதற்கு முன்னர் முதலில் ஜிம் சென்று அவரவர் உடல்களை தயார் செய்வார்கள். அதே போல Big Data, Machine Learning, Data Analytics, IOT போன்ற பல ஜொலிக்கும் துறைகளில் பயன்படுத்தப்படும் தரவுகள் நேரடியாக களம் இறக்கப்படுவதற்கு முன்னர் அதனை நேர்த்தியாக வடிவமைக்க பாண்டாஸ் உதவுகிறது.

அதாவது பல்வேறு வடிவ அமைப்புகளில் ஜிம்முக்கு வரும் பெண்களை பிரத்யேக வழிமுறைகளைக் கொண்டு கையாண்டு தகுந்த உடற்கட்டுடன் அவரவர் விரும்பும் துறைகளுக்குச் செல்ல உதவும் ஜிம் போல, பல்வேறு வடிவ அமைப்புகளில் வரும் தரவுகளை பிரத்யேக வழிமுறைகளைக் கொண்டு கையாண்டு நாம்

பயன்படுத்துவதற்குத் தக்க வடிவமைப்பில் மாற்றி அளிக்க இத்தகைய பாண்டாஸ் பயன்படுகிறது.

ஜிம் என்பது எவ்வாறு ஒவ்வொருவருடைய வடிவ அமைப்புக்கும் ஏற்றவாறு cardio, jumping jacks போன்ற ஒருசில பிரத்தியேக முறைகள் மூலம் அவரவர் விரும்பும் வடிவங்களான zerofit, biceps, muscle strengthening, body building போன்றவற்றைப் பெற உதவுகின்றதோ, அதே போல பாண்டாஸ் என்பதும் ஒவ்வொரு வகையான தரவின் வடிவ அமைப்புக்கும் ஏற்றவாறு (1D 2D or 3D data, time series data, categorical data like that) ஒருசில பிரத்தியேக முறைகள் மூலம் (Series, Data frame, Panel) நாம் பயன்படுத்துவதற்குத் தக்க வடிவமைப்பில் மாற்றி வழங்கப் பயன்படுகின்றது.

Big Data, Machine Learning, Data Analytics, IOT போன்ற துறைகளில் புழங்கப்படும் தரவுகளை எவ்வாறு pandas மூலம் கையாள்வது என்பதற்கு தேவையான அடிப்படைகளை இப்புத்தகம் எளிமையாக வழங்குகிறது.

தொடர்ந்து ஊக்கம் அளிக்கும் என் குடும்பத்தினருக்கும், கணியம் குழுவினருக்கும், FreeTamilEbooks.com குழுவினருக்கும், வாசகர்களுக்கும் நன்றிகள்.

து. நித்யா
மிசிசாகா, ஒன்றாரியோ, கனடா
16 செப்டம்பர் 2023

மின்னஞ்சல்: nithyadurai87@gmail.com

7

[illegible]: http://nithyashrinivasan.wordpress.com

உதாரணங்கள்

இந்த நூலில் உள்ள Pandas உதாரணங்கள் யாவும் https://gist.github.com/nithyadurai87 இங்கே உள்ளன.

பொருளடக்கம்

1 எளிய தமிழில் Pandas-1.....12
1.1 Data Types – Series, DataFrame, Panel.....13
2 எளிய தமிழில் Pandas-2.....20
2.1 Row & Column References.....20
3 எளிய தமிழில் Pandas-3.....25
3.1 DataFrame creation – Multiple ways.....25
3.1.1 From list of dicts.....26
3.1.3 From dict of lists.....28
3.1.4 From dict of series.....28
3.1.5 From csv file.....29
4 எளிய தமிழில் Pandas-4.....34
4.1 Attributes for Series, Dataframe, Panel.....34
5 எளிய தமிழில் Pandas-5.....43
5.1 Text Processing.....43
6 எளிய தமிழில் Pandas-6.....48
6.1 Location & Display properties.....48
7 எளிய தமிழில் Pandas-7.....55
7.1 SQL Operations.....55
7.1.1 Limit clause.....57
7.1.2 Where condition.....57
7.1.3 Grouping.....57
7.1.4 Combining data frames.....59
7.1.5 Joins.....61
7.1.6 Sorting.....62
8 எளிய தமிழில் Pandas-8.....63
8.1 Loops & Functions.....63
9 எளிய தமிழில் Pandas-9.....68
9.1 Metrics.....68
9.1.1 Percentage Change.....69
9.1.2 Covariance.....70
9.1.3 Correlation.....71
9.1.4 Ranks.....72
9.1.5 Rolling.....72
9.1.6 Expanding.....74
9.1.7 Exponential weighted functions.....75
10 எளிய தமிழில் Pandas-10.....77
10.1 Handling Null values.....77
11 எளிய தமிழில் Pandas-11.....81
11.1 Handling DateTime.....81
11.1.1 Supported format.....83
11.1.2 Timedelta.....84
11.1.3 Date Range.....85
11.1.4 Business dates.....86
11.1.5 Period Range.....87
11.1.6 DateTime & Timestamp.....87

12 எளிய தமிழில் Pandas-12..89 12.1 Handling Categorical data..89 13 எளிய தமிழில் Pandas-13_Final..96 13.1 Real-time Example..96 14 முடிவுரை..103 15 நூலாசிரியரின் பிற மின்னூல்கள்..104 16 கணியம் அறக்கட்டளை..105 16.1 தொலை நோக்கு - Vision..105 16.2 பணி இலக்கு - Mission..105

எளிய தமிழில் Pandas Kaniyam.com

1 எளிய தமிழில் Pandas-1

Pandas என்பது தரவுகளை வைத்து பல்வேறு ஆய்வினை நிகழ்த்துவதற்கு உதவும் வகையில் தரவினைப் பல்வேறு வடிவங்களில் சேமிக்கப் பயன்படுகிறது. Series, Dataframe, Panel ஆகியவை பாண்டாஸ் பயன்படுத்துகின்ற தரவு வடிவங்களாகும். இவை முறையே ஒருபரிமாண இருபரிமாண மற்றும் முப்பரிமாண வடிவில் அமையும் தரவுகளை சேமிக்கப் பயன்படுகின்றன. எடுத்துக்காட்டாக நமது அரசாங்கத்தில் என்னென்ன துறைகள் உள்ளன என்பதை வரிசையாக ஒன்றன்பின் ஒன்றாக எழுதி சேமிக்க series-ஐப் பயன்படுத்தலாம் (1D data). அதாவது துறைகள் எனும் இந்த ஒரு பரிமாணத்தில் மட்டும்தான் தரவுகள் சேமிக்கப்படுகின்றன. அதுவே ஒவ்வொரு துறையிலும் எவ்வளவு பேர் பணிபுரிகின்றனர் (2D data) என்பதை சேமிக்க விரும்பினால் dataframe-ஐப் பயன்படுத்தலாம். ஏனெனில் துறைகள், பணியாளர்கள் எனும் இரண்டு பரிமாணங்களில் தரவுகள் சேமிக்கப்படுகின்றன.

இன்னும் விரிவாக ஒவ்வொரு மாநிலத்திலும் உள்ள பல்வேறு துறைகளில் எத்தனை எத்தனை பேர் பணிபுரிகின்றனர் (3D data) என்பதை சேமிக்க விரும்பினால் Panel-ஐப் பயன்படுத்தலாம். ஏனெனில் மாநிலங்கள், அதில் உள்ள துறைகள், அவ்வொவ்வொன்றிலும் உள்ள பணியாளர்கள் எனும் மூன்று பரிமாணங்களில் இங்கு தரவுகள் சேமிக்கப்படும்.

மேலும் எண்கள், சொற்கள், categorical data, time series data போன்ற எந்த வகையான தரவுகளாக இருந்தாலும் அவற்றை நாம் சேமிக்கலாம். ஒவ்வொரு தரவு வடிவத்திற்கும் தனித்தனியான பண்புகள் உள்ளன. அவற்றை அணுகுவதற்கும் பயன்படுத்துவதற்கும் பல்வேறு வழிமுறைகளும் விதிமுறைகளும் உள்ளன. இவற்றைப் பற்றியெல்லாம் இப்புத்தகத்தில் காண்போம்.

முதலில் pandas பயன்படுத்தும் தரவு வடிவங்களைப் பற்றி விளக்கமாகக் காண்போம்.

1.1 Data Types – Series, DataFrame, Panel

Series என்பது ஒரே வகையான தரவினை மட்டும் சேமிக்கப் பயன்படுகிறது. எனவேதான் இது dtype எனும் பண்பினைப் பெற்று விளங்குகிறது.

Dataframe என்பது பல்வேறு வகையான தரவுகளை பல்வேறு சீரீஸில் சேமிக்கிறது. எனவே இதன் வடிவம் excel sheet-ல் உள்ளது போன்று row, column வடிவில் இருக்கும் . இதிலுள்ள ஒவ்வொரு column-ம் தனித்தனி சீரீஸ் ஆகும். ஒரு சீரீஸில் int வகையான மதிப்புகள் காணப்படும். அடுத்த சீரீஸில் string வகையான மதிப்புகள் இருக்கலாம். அதற்கடுத்தது datetime மதிப்பினைப் பெற்று விளங்கலாம். இதுபோன்று வெவ்வேறு வகையான மதிப்பினை பெற்று விளங்கும் பல்வேறு சீரீஸின் தொகுப்பே டேட்டாஃப்பிரேம் ஆகும். எனவேதான் இதற்கு dtype என்ற பொதுப்பண்பு கிடையாது. வேண்டுமானால் ஒவ்வொரு series-க்குமான dtype மதிப்பை இது தனித்தனியே வெளிப்படுத்தும்.

Panel என்பது இத்தகைய டேட்டாஃப்பிரேம்களின் தொகுப்பு ஆகும் . அதாவது ஒரு excelbook-ல் பல்வேறு sheets இருக்குமல்லவா! அந்த ஒவ்வொரு ஷீட்டிலும் rows, columns-ல் அமைந்த பல்வேறு தரவுகள் இருக்குமல்லவா! அதுபோலத்தான் பேனல் என்பது பல டேட்டாஃப்பிரேமை உள்ளடக்கியது. அதன் ஒவ்வொரு டேட்டாஃப்பிரேமும் rows, columns-ல் அமைந்த பல்வேறு தரவுகளைப் பெற்றிருக்கும்.

கீழ்க்கண்டவாறு நிரலை எழுதி இயக்கிப் பார்த்தால் இவற்றைப் பற்றிய புரிதல் உங்களுக்குத் தெளிவாகப் புலப்படும்.

```
import pandas as pd

print (pd.Serie s())

print (pd.DataFrame())

print (pd.Panel())



l1 = [90,83,67,83,45]

s = pd.Series(l1)

print (s)
```

```
df = pd.DataFrame(l1)

print (df)



l2 = [[90,83,67,83,45],[68,89,75,56,73],[58,88,60,90,100]]

df = pd.DataFrame(l2)

print (df)



d1 = {'Midterm': pd.DataFrame([[90,83,67,83,45],[68,89,75,56,73],
[58,88,60,90,100]]),

'Quarterly': pd.DataFrame([[35,44,65,56,79],[85,55,84,50,99],
[65,90,87,69,78]]),

'Half early': pd.DataFrame([[80,78,90,68,66],[35,89,67,79,90],
[67,89,59,90,45]]),

'Annual': pd.DataFrame([[90,94,58,69,84],[95,68,57,89,96],
[90,58,67,96,78]])}



p = pd.Panel(d1)

print (p)

print (p['Midterm'])

print (p.major_xs(1))

print (p.minor_xs(1))
```

```
print (s.dtype) # This attribute available only for series

print (df.dtypes)



print (df[1].dtype)
```

code link - 01_pandas_datatypes.py

நிரலுக்கான விளக்கம்:

எந்த ஒரு தகவலையும் பெற்றிராத காலியான சீரீஸ், டேட்டாஃப்பிரேம், பேனலின் வெளிப்பாடு பின்வருமாறு இருக்கும்.

```
print (pd.Series())
Series([], dtype: float64)


print (pd.DataFrame())
Empty DataFrame
Columns: []
Index: []


print (pd.Panel())
<class 'pandas.core.panel.Panel'>
Dimensions: 0 (items) x 0 (major_axis) x 0 (minor_axis)
Items axis: None
Major_axis axis: None
Minor_axis axis: None
```

ஒரு மாணவன் ஐந்து பாடத்திலும் பெற்ற மதிப்பெண்களை சேமிக்க இங்கு சீரீஸ் பயன்பட்டுள்ளது. அம்மதிப்பெண்கள் முதலில் ஒரு பட்டியலுக்குள் இடப்பட்டு, பின்னர் அப்பட்டியல் Series() -க்குள் செலுத்தப்பட்டுள்ளது. இதனை பிரிண்ட் செய்து பார்த்தால் அது பின்வருமாறு வெளிப்படுத்துவதைக் காணலாம். dtype

என்ற பண்பு இதில் int64 வகையிலான தரவுகள் சேமிக்கப் பட்டுள்ளன என்பதை வெளிப்படுத்தியுள்ளது.

```
l1 = [90,83,67,83,45]
s = pd.Series(l1)
print (s)


0 90
1 83
2 67
3 83
4 45
dtype: int64
```

அதேபோல மூன்று மாணவர்கள் ஐந்து பாடத்திலும் பெற்ற மதிப்பெண்களை சேமிக்க இங்கு டேட்டாஃப்பிரேம் பயன்பட்டுள்ளது. மூன்று மாணவர்களை rows-ஆகவும், ஐந்து பாடங்களை columns-ஆகவும் வைத்து இங்கு தரவுகள் சேமிக்கப்பட்டுள்ளன. rows என்பது index என்று அழைக்கப்படுகிறது. 0,1,2 ஆகியவை மூன்று மாணவர்களுக்கான இன்டெக்ஸ் ஆகும். 0,1,2,3,4 ஆகிய எண்களால் ஐந்து பாடங்களும் குறிக்கப்படுகின்றன. இந்த ஒவ்வொரு column-ம் ஒவ்வொரு தனித்தனி சீரீஸ் ஆகும்.

```
l2 = [[90,83,67,83,45],[68,89,75,56,73],[58,88,60,90,100]]
df = pd.DataFrame(l2)
print (df)




   0  1  2  3  4
0 90 83 67 83 45
1 68 89 75 56 73
2 58 88 60 90 100
```

அடுத்ததாக காலாண்டு, அரையாண்டு, முழு ஆண்டு என பல்வேறு பருவங்களில் இம்மூன்று மாணவர்களும் 5 பாடங்களில் பெற்ற மதிப்பெண்களை சேமிக்க பேனல் பயன்பட்டுள்ளது. இப்பருவங்களை keys-ஆகவும், ஒவ்வொரு பருவத்திலும்

அம்மூன்று மாணவர்களும் பெற்ற மதிப்பெண்களைக் கொண்ட டேட்டாஃப்பிரேமை values-ஆகவும் கொண்டு ஒரு டிக்சனரி உருவாக்கப்பட்டுள்ளது. பின்னர் அந்த டிக்சனரியை Panel-க்குள் செலுத்தி பிரிண்ட் செய்யப்படுகிறது.

சீரீஸ் , டேட்டாஃப்பிரேம் போன்று இதன் வெளிப்பாடு தரவுகளை மொத்தமாகக் காட்டாது. பேனலின் டைமென்ஷன் எத்தனை?, ஒவ்வொரு பரிமாணத்திலும் எத்தனை இன்டெக்ஸ் மற்றும் columns உள்ளன?, அவைகளின் range எதிலிருந்து எதுவரை அமைந்துள்ளன ஆகியவற்றை மட்டுமே வெளிப்படுத்தும். கீழே காலாண்டு, அரையாண்டு என்று வகைப்படுத்த கூடிய பாகுபாடு மொத்தம் 4 உள்ளது என்பதை வெளிப்படுத்தியுள்ளது. பின் அவை ஒவ்வொன்றிலும் சேமிக்கப்பட்டுள்ள மாணவர்கள் 3 என்பதை மேஜர் ஆக்சிஸ் ஆகவும், பாடங்கள் 5 என்பதை மைனர் ஆக்சிஸ் ஆகவும் வெளிப்படுத்தியுள்ளது. அடுத்ததாக அவைகள் ஒவ்வொன்றும் எதிலிருந்து எதுவரை அமைந்துள்ளன எனும் range-ஐ வெளிப்படுத்தியுள்ளது.

```
d1 = {'Midterm': pd.DataFrame([[90,83,67,83,45],[68,89,75,56,73],
[58,88,60,90,100]]),
'Quarterly': pd.DataFrame([[35,44,65,56,79],[85,55,84,50,99],
[65,90,87,69,78]]),
'Half early': pd.DataFrame([[80,78,90,68,66],[35,89,67,79,90],
[67,89,59,90,45]]),
'Annual': pd.DataFrame([[90,94,58,69,84],[95,68,57,89,96],
[90,58,67,96,78]])}




p = pd.Panel(d1)



print (p)


<class 'pandas.core.panel.Panel'>
Dimensions: 4 (items) x 3 (major_axis) x 5 (minor_axis)
Items axis: Midterm to Annual
Major_axis axis: 0 to 2
Minor_axis axis: 0 to 4
```

பேனலின் ஏதாவது ஒரு item-ஐ மட்டும் தேர்வு செய்து பார்க்க விரும்பினால் அது பின்வருமாறு அமையும்.

```
print (p['Midterm'])


   0  1  2  3  4
0 90 83 67 83 45
1 68 89 75 56 73
2 58 88 60 90 100
```

ஒவ்வொரு பருவத்திலும் இரண்டாவது மாணவன் மட்டும் பெற்ற மதிப்பெண்களைக் காண விரும்பினால், மேஜர் ஆக்சிஸ் என்பதற்குள் இரண்டாவது மாணவனுக்கான இன்டெக்ஸ் எண்ணைக் கொடுத்து பிரிண்ட் செய்து பார்க்கவும்.

```
print (p.major_xs(1))


Midterm Quarterly Half early Annual
0 68      85         35          95
1 89      55         89          68
2 75      84         67          57
3 56      50         79          89
4 73      99         90          96
```

அதே போல ஒவ்வொரு பருவத்திலும் இரண்டாவது பாடத்தில் மட்டும் பெற்ற மதிப்பெண்களைக் காண விரும்பினால், மைனர் ஆக்சிஸ் என்பதற்குள் இரண்டாவது பாடத்திற்கான இன்டெக்ஸ் எண்ணை மட்டும் கொடுத்து பிரிண்ட் செய்து பார்க்கவும் .

```
print (p.minor_xs(1))


Midterm Quarterly Half early Annual
0 83      44         78          94
1 89      55         89          68
2 88      90         89          58
```

2 எளிய தமிழில் Pandas-2

2.1 Row & Column References

மூன்று மாணவர்கள் மற்றும் ஐந்து பாடங்களை வெறும் எண்களால் குறிப்பிடாமல் அவற்றுக்கான பெயர்களை வைத்துக் குறிப்பிட்டால் அணுகுவதற்கும் , புரிந்துகொள்வதற்கும் இன்னும் சுலபமாக இருக்கும் அல்லவா ? அதற்காகத்தான் இன்டெக்ஸ் மற்றும் columns ஆகிய பண்புகள் பயன்படுகின்றன.

இவைகளைப் பயன்படுத்தி எழுதப்பட்டுள்ள நிரல் பின்வருமாறு.

```
import pandas as pd

l1 = [90,83,67,83,45]

l2 = [[90,83,67,83,45],[68,89,75,56,73],[58,88,60,90,100]]



df = pd.DataFrame(l2)

print (df)

print (df[2])

print (df[:2])

print (df[-2:])



df =
pd.DataFrame(l2,columns=['Tamil','English','Maths','Science','Social'
],index=['Ramesh','Suresh','Kamesh'],dtype='int32')
```

```
print (df)

print (df['Maths'])

print (df[:'Suresh'])

print (df['Suresh':])



s =
pd.Series(l1,index=['Tamil','English','Maths','Science','Social'])

print (s[['Tamil','Social']])
```

code link – 02_pandas_rowcolumnreferences.py

நிரலுக்கான விளக்கம்:

முதலில் பின்வருமாறு ஒரு டேட்டாஃப்பிரேமை உருவாக்கிக் கொள்ளவும்.

```
l2 = [[90,83,67,83,45],[68,89,75,56,73],[58,88,60,90,100]]
df = pd.DataFrame(l2)
print (df)


   0  1  2  3  4
0 90 83 67 83 45
1 68 89 75 56 73
2 58 88 60 90 100
```

df[2] என்பது 2 எனும் column எண்ணைக் கொண்ட சீரீசை வெளிப்படுத்தும்.

```
print (df[2])


0 67
1 75
2 60
Name: 2, dtype: int64
```

df[:2] என்பது colon-க்கு முன்னால் ஏதேனும் எண் குறிப்பிடப்பட்டிருப்பின் அதிலிருந்து தொடங்கி, colon-க்கு அடுத்து குறிப்பிடப்பட்டுள்ள எண்ணுக்கு முந்தைய எண் வரை இருக்கும் பெயர்களைக் கொண்ட rows-ஐ வெளிப்படுத்தும்.

```
print (df[:2])


   0  1  2  3  4
0 90 83 67 83 45
1 68 89 75 56 73
```

df[-2:] என்பது கீழிருந்து மேல் வாக்கில் கொடுக்கப்பட்டுள்ள எண்ணுக்கு ஏற்றவாறு rows-ஐ வெளிப்படுத்தும்.

```
print (df[-2:])


   0  1  2  3  4
1 68 89 75 56 73
2 58 88 60 90 100
```

இவ்வாறு வெறும் எண்களை வைத்து அணுகாமல், ஒரு டேட்டாஃப்பிரேமை உருவாகும்போதே என்னென்ன row-க்கு என்னென்ன பெயர் மற்றும் என்னென்ன column-க்கு என்னென்ன பெயர் என்பதைக் கொடுத்து உருவாக்கினால் நமக்கு வேண்டிய rows மற்றும் columns-ஐ எடுப்பதற்கும் அணுகுவதற்கும் சுலபமாக இருக்கும். rows-க்கான பெயர்களைக் குறிப்பிட இன்டெக்ஸ் பயன்படுகிறது.

columns-□□□□□ □□□□□□□□□□ □□□□□□□□□ columns □□□□□□□□□□□□. □□□ □□□□□□□□□□□□.

```
df =
pd.DataFrame(l2,columns=['Tamil','English','Maths','Science','Social'
],index=['Ramesh','Suresh','Kamesh'],dtype='int32′)


print (df)


       Tamil English Maths Science Social
Ramesh 90    83      67    83      45
Suresh 68    89      75    56      73
Kamesh 58    88      60    90      100
```

df['Maths'] □□□□□□ Maths □□□□□ □□□□□□□□ □□□□□ column-□ □□□□□□□□□□□□□□□□□□□□□□.

```
print (df['Maths'])


Ramesh 67
Suresh 75
Kamesh 60
Name: Maths, dtype: int32
```

df[:'Suresh'] □□□□□□□ colon-□□□□ □□□□ □□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□ □□□□□□□□ □□□□□□□ □□□□□□□□ □□□□□□□□ □□□□□ row □□□□□ □□□ □□□□□□□□□□□□□□□□□□□□□□.

```
print (df[:'Suresh'])


       Tamil English Maths Science Social
Ramesh 90    83      67    83      45
Suresh 68    89      75    56      73
```

df['Suresh':] என்பது colon-க்கு அடுத்து எதுவும் கொடுக்கப்படவில்லையாதலால், சுரேஷ் இலிருந்து துவங்கி கடைசி row வரை வெளிப்படுத்தியுள்ளது.

```
print (df['Suresh':])


       Tamil English Maths Science Social
Suresh 68    89      75    56      73
Kamesh 58    88      60    90      100
```

ஆனால் சீரீஸ் உருவாக்கும் போது தரவுகளுக்குப் பெயர் கொடுக்க இன்டெக்ஸ் என்ற பண்பு மட்டுமே பயன்படும். ஏனெனில் இது ஒரு பரிமாணத் தரவுகள் ஆதலால் columns என்ற பண்பு இராது.

```
l1 = [90,83,67,83,45]
s =
pd.Series(l1,index=['Tamil','English','Maths','Science','Social'])
print (s[['Tamil','Social']])


Tamil  90
Social 45
dtype: int64
```

3 எளிய தமிழில் Pandas-3

3.1 DataFrame creation – Multiple ways

ஒரு டேட்டாஃப்ப்பிரேமை பல்வேறு வழிகளில் உருவாக்கலாம். அறிமுகத்தின் போது ஒரு லிஸ்ட் உள்ளே பல லிஸ்டை கொடுத்து உருவாக்கினோம் அல்லவா ! அதேபோல இன்னும் என்னென்ன வழிகளில் எல்லாம் உருவாக்கலாம் என்பதை இப்பகுதியில் காணலாம்.

அவை பின்வருமாறு.

```
import pandas as pd

d = {'Tamil' : 90, 'English' : 83, 'Maths' : 67, 'Science' : 83,
'Social' : 45}

s = pd.Series(d)

print (s)



l = [{'Tamil' : 90, 'English' : 83, 'Maths' : 67, 'Science' : 83,
'Social' : 45},
{'Tamil' : 68, 'English' : 89, 'Maths' : 75, 'Science' : 56, 'Social'
: 73},
{'Tamil' : 58, 'English' : 88, 'Maths' : 60, 'Science' : 90, 'Social'
: 100}]



df = pd.DataFrame(l)

print (df)

df = df[['Tamil','English','Maths','Science','Social']]
```

```
print (df)

d = {'Tamil' : [90,68,58], 'English' : [83,89,88], 'Maths' :
[67,75,60], 'Science' : [83,56,90], 'Social' : [45,73,100]}

df = pd.DataFrame(d)

print (df)

d = {'Tamil' : pd.Series([90,68,58]), 'English' :
pd.Series([83,89,88]), 'Maths' : pd.Series([67,75,60]), 'Science' :
pd.Series([83,56,90]), 'Social' : pd.Series([45,73,100])}

df = pd.DataFrame(d)

print (df)

df['Environmental Science']=pd.Series([90,92,98])

print (df)

df = pd.DataFrame(d,columns=['Tamil','English','Social'])

print (df)
```

code link - 02_pandas_rowcolumnreferences.py

3.1.1 From list of dicts

[illegible]

[illegible]

```
d = {'Tamil' : 90, 'English' : 83, 'Maths' : 67, 'Science' : 83,
'Social' : 45}
s = pd.Series(d)
print (s)


Tamil   90
English 83
Maths   67
Science 83
Social  45
dtype: int64
```

மேற்கூறியவாறே பல டிக்சனரிகளை ஒரு லிஸ்டுக்குள் கொடுத்து டேட்டாஃப்பிரேம் உருவாக்கிக் காட்டப்பட்டுள்ளது .

```
l = [{'Tamil' : 90, 'English' : 83, 'Maths' : 67, 'Science' : 83,
'Social' : 45},
{'Tamil' : 68, 'English' : 89, 'Maths' : 75, 'Science' : 56, 'Social'
: 73},
{'Tamil' : 58, 'English' : 88, 'Maths' : 60, 'Science' : 90, 'Social'
: 100}]
df = pd.DataFrame(l)
print (df)


  English Maths Science Social Tamil
0 83      67    83      45     90
1 89      75    56      73     68
2 88      60    90      100    58
```

ஒரு லிஸ்டுக்குள் இருக்கும் டிக்சனரியின் கீஸ் column-ஆக மாறும்போது அது ஆல்பபெட் முறையில் columns-ஐ அமைக்கும். அதனை நாம் விரும்புகின்ற வரிசையில் அமைக்க விரும்பினால் பின்வருமாறு கொடுக்க வேண்டும் .

```
df = df[['Tamil','English','Maths','Science','Social']]
print (df)


  Tamil English Maths Science Social
0 90    83      67    83      45
1 68    89      75    56      73
2 58    88      60    90      100
```

3.1.2

3.1.3 From dict of lists

மேற்கூறியவாறு அல்லாமல் நேரடியாக டிக்சனரியின் key-ஆக columns-ஐயும், அதன் மதிப்புகளை ஒரு லிஸ்ட் ஆகவும் கொடுத்து டேட்டாஃப்பிரேமை உருவாக்கினால், column வரிசையை மாற்ற வேண்டிய அவசியம் இருக்காது. ஏனெனில் நாம் கொடுக்கின்ற வரிசையிலே column அமைக்கப்படும். இது பின்வருமாறு.

```
d = {'Tamil' : [90,68,58], 'English' : [83,89,88], 'Maths' :
[67,75,60], 'Science' : [83,56,90], 'Social' : [45,73,100]}
df = pd.DataFrame(d)
print (df)


  Tamil English Maths Science Social
0 90    83      67    83      45
1 68    89      75    56      73
2 58    88      60    90      100
```

3.1.4 From dict of series

அதேபோல ஒரு டிக்சனரியின் கீ மதிப்புகளை லிஸ்ட் ஆக அல்லாமல் சீரீஸ் ஆக கொடுத்தும் டேட்டாஃப்பிரேமை உருவாக்கலாம். இது பின்வருமாறு.

```
d = {'Tamil' : pd.Series([90,68,58]), 'English' :
pd.Series([83,89,88]), 'Maths' : pd.Series([67,75,60]),'Science' :
pd.Series([83,56,90]), 'Social' : pd.Series([45,73,100])}
df = pd.DataFrame(d)
print (df)
```

```
  Tamil English Maths Science Social
0 90    83      67    83      45
1 68    89      75    56      73
2 58    88      60    90      100
```

புதிதாக ஒரு column-ஐ இணைக்க விரும்பினால் அது பின்வருமாறு அமையும்.

```
df['Environmental Science']=pd.Series([90,92,98])
print (df)


  Tamil English Maths Science Social Environmental Science
0 90    83      67    83      45     90
1 68    89      75    56      73     92
2 58    88      60    90      100    98
```

ஒரு டிக்சனரியிலிருந்து நமக்கு வேண்டிய column-ஐ மட்டும் எடுத்து டேட்டாஃப்பிரேமை உருவாக்க விரும்பினால் அது பின்வருமாறு அமையும்.

```
df = pd.DataFrame(d,columns=['Tamil','English','Social'])
print (df)


  Tamil English Social
0 90    83      45
1 68    89      73
2 58    88      100
```

3.1.5 From csv file

ஒரு சி.எஸ்.வி கோப்பில் இருக்கும் தரவுகளை வைத்து டேட்டாஃப்பிரேமை உருவாக்குவதில் பலவிதங்கள் உள்ளன. அவை பின்வருமாறு.

```
import pandas as pd
```

```
import numpy as np


df = pd.read_csv("./girls.csv")

print (df)


df = pd.read_csv("./girls.csv",index_col=['id'])

print (df)


df = pd.read_csv("./girls.csv",names=['a', 'b', 'c','d','e','f','g'])

print (df)


df=pd.read_csv("./girls.csv",header=4)

print (df)


df=pd.read_csv("./girls.csv",skiprows=5,dtype={'40': np.float64})

print (df)

df.to_csv('aaa.csv')
```

Code link - view raw 03b_pandas_dfcreation.py

pd.read_csv() [illegible] function [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. [illegible] [illegible] [illegible] [illegible] [illegible] row-[illegible] [illegible] [illegible] [illegible] [illegible] index [illegible] [illegible]. [illegible] [illegible] row [illegible] column [illegible] [illegible].

```
df = pd.read_csv(“./girls.csv")
print (df)

  id fname    lname     age desig      no         place
0 1  Nithya   Duraisamy 31  Manager    9587412536 Hyderabad
1 2  Nandhini Babu      28  AstManager 9848022338 Delhi
2 3  Madhuri  Nathan    51  VP         9848022339 Delhi
3 4  Kavitha  Manoharan 45  AVP        9848022330 Hyderabad
4 5  Vijaya   Kandasamy 40  AVP        9848022336 Noida
5 6  Aarthi   Raj       22  AstManager 9848022335 Chennai
6 7  Lavanya  Sankar    23  SrEngineer 9848022334 Chennai
7 8  Meena    Baskar    56  VP         9848022333 Hyderabad
8 9  Gayathri Ragu      36  Engineer   9848022333 Chennai
9 10 Kavitha  Manoharan 49  AVP        9848022336 Noida
```

[illegible] [illegible] [illegible] [illegible] column-[illegible] [illegible] [illegible] [illegible] [illegible] index_col [illegible] [illegible] [illegible].

```
df = pd.read_csv(“./girls.csv",index_col=['id'])
print (df)

id fname    lname     age desig      no         place
1  Nithya   Duraisamy 31  Manager    9587412536 Hyderabad
2  Nandhini Babu      28  AstManager 9848022338 Delhi
3  Madhuri  Nathan    51  VP         9848022339 Delhi
4  Kavitha  Manoharan 45  AVP        9848022330 Hyderabad
5  Vijaya   Kandasamy 40  AVP        9848022336 Noida
6  Aarthi   Raj       22  AstManager 9848022335 Chennai
7  Lavanya  Sankar    23  SrEngineer 9848022334 Chennai
8  Meena    Baskar    56  VP         9848022333 Hyderabad
9  Gayathri Ragu      36  Engineer   9848022333 Chennai
10 Kavitha  Manoharan 49  AVP        9848022336 Noida
```

default-[illegible] row-[illegible], [illegible] names [illegible].

```
df = pd.read_csv(“./girls.csv",names=['a', 'b',
'c','d','e','f','g'])
print (df)

   a    b        c         d   e           f          g
0  id   fname    lname     age desig       no         place
1  001  Nithya   Duraisamy 31  Manager     9587412536 Hyderabad
2  002  Nandhini Babu      28  AstManager  9848022338 Delhi
3  003  Madhuri  Nathan    51  VP          9848022339 Delhi
4  004  Kavitha  Manoharan 45  AVP         9848022330 Hyderabad
5  005  Vijaya   Kandasamy 40  AVP         9848022336 Noida
6  006  Aarthi   Raj       22  AstManager  9848022335 Chennai
7  007  Lavanya  Sankar    23  SrEngineer  9848022334 Chennai
8  008  Meena    Baskar    56  VP          9848022333 Hyderabad
9  009  Gayathri Ragu      36  Engineer    9848022333 Chennai
10 010  Kavitha  Manoharan 49  AVP         9848022336 Noida
```

default-[illegible] row-[illegible], [illegible] row-[illegible] column-[illegible] header [illegible]. [illegible] row-[illegible] rows [illegible].

```
df=pd.read_csv(“./girls.csv",header=4)
print (df)

   004  Kavitha  Manoharan 45  AVP         9848022330 Hyderabad
0  5    Vijaya   Kandasamy 40  AVP         9848022336 Noida
1  6    Aarthi   Raj       22  AstManager  9848022335 Chennai
2  7    Lavanya  Sankar    23  SrEngineer  9848022334 Chennai
3  8    Meena    Baskar    56  VP          9848022333 Hyderabad
4  9    Gayathri Ragu      36  Engineer    9848022333 Chennai
5  10   Kavitha  Manoharan 49  AVP         9848022336 Noida
```

அதேபோல skiprows எனும் பண்பும் கொடுக்கப்பட்டுள்ள எண்ணிக்கைக்கு ஏற்ப ஆரம்பத்தில் இருக்கும் rows-ஐ தவிர்த்து டேட்டாஃப்பிரேமை உருவாக்கும். dtype எனும் பண்பு குறிப்பிடப்பட்டுள்ள column இன் தரவு வகையை மாற்றப் பயன்படும்.

```
df=pd.read_csv(“./girls.csv",skiprows=5,dtype={'40': np.float64})
print (df)


   005  Vijaya   Kandasamy 40    AVP        9848022336 Noida
0  6    Aarthi   Raj       22.0  AstManager 9848022335 Chennai
1  7    Lavanya  Sankar    23.0  SrEngineer 9848022334 Chennai
2  8    Meena    Baskar    56.0  VP         9848022333 Hyderabad
3  9    Gayathri Ragu      36.0  Engineer   9848022333 Chennai
4  10   Kavitha  Manoharan 49.0  AVP        9848022336 Noida
```

4 எளிய தமிழில் Pandas-4

4.1 Attributes for Series, Dataframe, Panel

பாண்டாஸ் ஆதரிக்கும் இம்மூன்று தரவு வகைகளுக்குமான பண்புகள் என்னென்ன செய்திகளை வெளிப்படுத்துகின்றன என்பது பின்வருமாறு கொடுக்கப்பட்டுள்ளது.

```
import pandas as pd



s = pd.Series([90,83,67,83,45])

df = pd.DataFrame([[90,83,67,83,45],[68,89,75,56,73],
[58,88,60,90,100]])

p = pd.Panel({'Midterm': pd.DataFrame([[90,83,67,83,45],
[68,89,75,56,73],[58,88,60,90,100]]),

'Quarterly': pd.DataFrame([[35,44,65,56,79],[85,55,84,50,99],
[65,90,87,69,78]])})



print (s.axes)

print (df.axes)

print (p.axes)



df2 = pd.DataFrame()

print (s.empty)
```

```
print (df2.empty)

print (p.empty)



print (s.shape)

print (df.shape)

print (p.shape)



print (s.ndim)

print (df.ndim)

print (p.ndim)



print (s.size)

print (df.size)

print (p.size)



print (s)

print (s.values)

print (df.values)

print (p.values)
```

```
print (s.head(2))

print (df.head(2))

print (s.tail(2))

print (df.tail(2))

# panel doesn't have these attributes



print (df.T)

# Panel & Series doesn't have these attributes



print (s.dtypes)

print (df.dtypes)

print (p.dtypes)



print (df)

print (df.describe())

print (df.sum())

print (df.sum(1))

print (df.mean(1))
```

```
print (df.std(1))
```

Code link - [view raw](#) [04_pandas_attributes.py](#)

[illegible].

```
s = pd.Series([90,83,67,83,45])


df = pd.DataFrame([[90,83,67,83,45],[68,89,75,56,73],
[58,88,60,90,100]])


p = pd.Panel({'Midterm': pd.DataFrame([[90,83,67,83,45],
[68,89,75,56,73],[58,88,60,90,100]]),
'Quarterly': pd.DataFrame([[35,44,65,56,79],[85,55,84,50,99],
[65,90,87,69,78]])})
```

Axes [illegible].

```
print (s.axes)

[RangeIndex(start=0, stop=5, step=1)]


print (df.axes)
[RangeIndex(start=0, stop=3, step=1), RangeIndex(start=0, stop=5,
step=1)]


print (p.axes)
[Index(['Midterm', 'Quarterly'], dtype='object'), RangeIndex(start=0,
stop=3, step=1), RangeIndex(start=0, stop=
5, step=1)]
```

Empty எனும் பண்பு தரவுகள் ஏதேனும் சேமிக்கப்பட்டுள்ளதா இல்லையா என்பதை வெளிப்படுத்துகிறது.

```
df2 = pd.DataFrame()
print (s.empty)
False
print (df2.empty)
True
print (p.empty)
False
```

shape என்பது ஒவ்வொரு பரிமாணத்திலும் சேமிக்கப்பட்டுள்ள தரவுகளின் வடிவத்தை வெளிப்படுத்தும்.

```
print (s.shape)
(5,)
print (df.shape)
(3, 5)
print (p.shape)
(2, 3, 5)
```

ndim என்பது எத்தனை பரிமாணங்கள் என்பதை வெளிப்படுத்தும் .

```
print (s.ndim)
1
print (df.ndim)
2
print (p.ndim)
3
```

Size என்பது ஒவ்வொன்றிலும் மொத்தமுள்ள தரவுகளின் அளவை வெளிப்படுத்தும்.

```
print (s.size)
5
print (df.size)
15
print (p.size)
30
```

values என்பது ஒவ்வொன்றிலும் சேமிக்கப்பட்டுள்ள தரவுகளை எடுத்து வெளிக்காட்டும்.

```
print (s)
0 90
1 83
2 67
3 83
4 45
dtype: int64

print (s.values)
[90 83 67 83 45]

print (df.values)
[[ 90 83 67 83 45]
[ 68 89 75 56 73]
[ 58 88 60 90 100]]

print (p.values)
[[[ 90 83 67 83 45]
[ 68 89 75 56 73]
[ 58 88 60 90 100]]
[[ 35 44 65 56 79]
[ 85 55 84 50 99]
[ 65 90 87 69 78]]]
```

முதலிரண்டு அல்லது கடைசியில் இருந்து சில என்பது போன்று எடுத்துப் பார்க்க விரும்பினால் head / tail ஆகியவை பயன்படும். பேனலுக்கு இந்த பண்பு கிடையாது.

```
print (s.head(2))
0 90
1 83
dtype: int64

print (df.head(2))
  0  1  2  3  4
0 90 83 67 83 45
1 68 89 75 56 73

print (s.tail(2))
3 83
4 45
dtype: int64
```

```
print (df.tail(2))
  0  1  2  3  4
1 68 89 75 56 73
2 58 88 60 90 100


# panel doesn't have these attributes
```

[illegible] Transpose [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

```
print (df.T)
  0  1  2
0 90 68 58
1 83 89 88
2 67 75 60
3 83 56 90
4 45 73 100


# Panel & Series doesn't have these attributes
```

dtypes [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

```
print (s.dtypes)
int64


print (df.dtypes)
0 int64
1 int64
2 int64
3 int64
4 int64
dtype: object


print (p.dtypes)
Midterm int64
Quarterly int64
dtype: object
```

```
print (df)
  0  1  2  3  4
0 90 83 67 83 45
1 68 89 75 56 73
2 58 88 60 90 100
```

describe() எனும் பண்பு அடிப்படையான ஒரு சில புள்ளியியல் விவரங்களைக் கணக்கிட்டு வெளிப்படுத்தப் பயன்படுகிறது. ஒவ்வொரு Column-லும் சேமிக்கப்பட்டுள்ள தரவுகளுக்கான count, mean, standard deviation, minimum, maximum மதிப்புகள் ஆகியவற்றை வெளிப்படுத்தியுள்ளது. இங்கு மொத்தமே மூன்று தரவுகள் உள்ளதால் 25%, 50%,75% எனத் தரவுகளைப் பிரித்து வெளிப்படுத்துவதன் முக்கியத்துவத்தை நம்மால் சரிவர உள்வாங்கிக் கொள்ள முடியவில்லை. ஆனால் அதிக அளவு தரவுகளை நிகழ் நேரத்தில் கையாளும்போது இதுபோன்ற புள்ளிவிவரங்கள் பயனுள்ளதாக இருக்கும்.

```
print (df.describe())


       0         1          2          3          4
count 3.000000  3.000000  3.000000  3.000000  3.000000
mean  72.000000 86.666667 67.333333 76.333333 72.666667
std   16.370706 3.214550  7.505553  17.953644 27.501515
min   58.000000 83.000000 60.000000 56.000000 45.000000
25%   63.000000 85.500000 63.500000 69.500000 59.000000
50%   68.000000 88.000000 67.000000 83.000000 73.000000
75%   79.000000 88.500000 71.000000 86.500000 86.500000
max   90.000000 89.000000 75.000000 90.000000 100.000000
```

sum() என்பது ஒவ்வொரு பாடத்திலும் மூன்று வெவ்வேறு மாணவர்கள் பெற்ற மதிப்பெண்களை கூட்டிக் காட்டியுள்ளது. ஆனால் இது போன்று நாம் கூட்ட மாட்டோம் அல்லவா! ஒவ்வொரு மாணவனும் ஐந்து பாடத்தில் பெற்ற மதிப்பெண்களைக் கூட்டுமாறு செய்ய sum(1) அல்லது sum(axis=1) எனக் கொடுக்க வேண்டும். axis=0 என்பது row-ஐயும், 1 என்பது column-ஐயும் குறிக்கும் .

```
print (df.sum())
0 216
1 260
2 202
3 229
```

```
4 218
dtype: int64


print (df.sum(1))
0 368
1 361
2 396
dtype: int64
```

அதேபோல mean(), std() ஆகியவற்றையும் தனித்தனியே கணக்கிடலாம்.

```
print (df.mean(1))
0 73.6
1 72.2
2 79.2
dtype: float64


print (df.std(1))
0 18.077610
1 11.945711
2 19.005262
dtype: float64
```

5 □□□□□ □□□□□□□□□ Pandas-5

5.1 Text Processing

□□□ □□□□□□□□□□□□□□□□ / □□□□□□□□□ □□□□□□□□□□□□□□□□□□ □□□□□□□□□ □□□□□□□□□ □□□□□□□□ □□□□□ □□□□□□□□□□□ □□□□□□□□□ functions-□ □□□□□□□□□□□□ □□□□□□□. □□□□□□□ □□□□□□□□□ □□□□□□□□□ □□□□□□□□□ □□□□□□□□□ □□□□□□□□□□□. □□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□, □□□□□□□□□□□ □□□ □□□□□□ □□□□□□□, □□□□ □□□□□□□□ □□□□□□□□□ functions-□ □□□□□ □□□□□ □□□□□□□□.

□□□□□□□□ □□□□□□□□□ □□□□□□□□□□□□.

```
import pandas as pd

df = pd.DataFrame({'Languages': pd.Series(['Tamil','English']),
'Subjects': pd.Series(['Maths','Science','Social'])})

print (df)

print (df['Languages'].str.upper()) # lower(), swapcase(),
islower(), isupper(), isnumeric()

print (df['Languages'].str.split('i'))

print (df['Languages'].str.contains('i'))

print (df['Languages'].str.len())

print (df['Subjects'].str.startswith('S')) # endswith()

print (df['Subjects'].str.count('e'))

print (df['Subjects'].str.find('e'))

print (df['Subjects'].str.findall('e'))

print (df['Languages'].str.replace('i','e'))
```

```
print (df['Languages'].str.cat(sep=','))



s = pd.Series(['Ramu','Somu','Jothi','Rathi','Jothi'])

print (s.str.get_dummies())

print (s.str.repeat(4))
```

Code link - view raw 05_pandas_texts.py

[illegible], [illegible].

```
df = pd.DataFrame({'Languages': pd.Series(['Tamil','English']),
'Subjects': pd.Series(['Maths','Science','Social'])})



print (df)
Languages Subjects
0 Tamil Maths
1 English Science
2 NaN Social
```

[illegible] Language [illegible], [illegible]. [illegible].

```
print (df['Languages'].str.upper()) # lower(), swapcase(), islower(),
isupper(), isnumeric()
0 TAMIL
1 ENGLISH
2 NaN
Name: Languages, dtype: object
```

ஏதேனும் ஒரு குறிப்பிட்ட எழுத்து அல்லது குறியீடு மூலம் சொற்களைப் பிரித்து பல்வேறு வார்த்தைகளாக ஒரு லிஸ்ட் உள் அமைக்க split() பயன்படுகிறது.

```
print (df['Languages'].str.split('i'))
0 [Tam, l]
1 [Engl, sh]
2 NaN
Name: Languages, dtype: object
```

ஏதேனும் ஒரு குறிப்பிட்ட எழுத்து அல்லது குறியீடு நமது சீரீஸில் உள்ளதா இல்லையா என்பதைக் கண்டுபிடிக்க contains () பயன்படுகிறது.

```
print (df['Languages'].str.contains('i'))
0 True
1 True
2 NaN
Name: Languages, dtype: object
```

நமது சீரீஸில் சேமிக்கப்பட்டுள்ள சொற்களின் நீளங்களைக் கண்டுபிடிக்க len() பயன்படுகிறது.

```
print (df['Languages'].str.len())
0 5.0
1 7.0
2 NaN
Name: Languages, dtype: float64
```

ஒரு குறிப்பிட்ட எழுத்தில் சொற்கள் தொடங்கியுள்ளதா இல்லையா என்பதைக் கண்டுபிடிக்க startswith() பயன்படுகிறது.

```
print (df['Subjects'].str.startswith('S')) # endswith()
0 False
1 True
2 True
Name: Subjects, dtype: bool
```

ஒரு குறிப்பிட்ட எழுத்து ஒரு சொல்லில் எத்தனை முறை வந்துள்ளது என்பதைக் கண்டுபிடிக்க count() பயன்படுகிறது.

```
print (df['Subjects'].str.count('e'))
0 0
```

```
1 2
2 0
Name: Subjects, dtype: int64
```

ஒரு குறிப்பிட்ட எழுத்து ஒரு சொல்லில் எந்த இடத்தில் அமைந்துள்ளது என்பதைக் கண்டுபிடிக்க find() பயன்படுகிறது. இதன் வெளியீடு 0 முதல் அதன் நீளத்திற்கு ஏற்ப அமையும். -1 என்றால் அந்த எழுத்து எவ்விடத்திலும் அமையவில்லை என்று பொருள்.

```
print (df['Subjects'].str.find('e'))
0 -1
1 3
2 -1
Name: Subjects, dtype: int64


print (df['Subjects'].str.findall('e'))
0 []
1 [e, e]
2 []
Name: Subjects, dtype: object
```

ஒரு எழுத்தை மற்றொரு எழுத்தால் இடமாற்றம் செய்ய replace பயன்படுகிறது.

```
print (df['Languages'].str.replace('i','e'))
0 Tamel
1 Englesh
2 NaN
Name: Languages, dtype: object
```

சீரீஸில் சேமிக்கப்பட்டுள்ள சொற்களை ஏதோ ஒரு குறியீடு கொண்டு பிரித்து வெளிப்படுத்த separator பயன்படுகிறது.

```
print (df['Languages'].str.cat(sep=','))
Tamil,English
```

ஒரு சீரீஸில் உள்ள unique சொற்களை எடுத்து அவை ஒவ்வொன்றும் எத்தனை முறை வந்துள்ளன என்பதைக் கண்டுபிடிக்க get_dummies பயன்படும்.

```
s = pd.Series(['Ramu','Somu','Jothi','Rathi','Jothi'])
```

```
print (s.str.get_dummies())
Jothi Ramu Rathi Somu
0 0 1 0 0
1 0 0 0 1
2 1 0 0 0
3 0 0 1 0
4 1 0 0 0
```

[illegible] repeat() [illegible].

```
print (s.str.repeat(4))
0 RamuRamuRamuRamu
1 SomuSomuSomuSomu
2 JothiJothiJothiJothi
3 RathiRathiRathiRathi
4 JothiJothiJothiJothi
dtype: object
```

6 எளிய தமிழில் Pandas-6

6.1 Location & Display properties

ஒரு டேட்டாஃப்பிரேமில் இருப்பவற்றை கொஞ்சம் கொஞ்சமாக நமது தேவைக்கு ஏற்றவாறு வெட்டி எடுத்து, பிரித்துப் பார்த்து புரிந்து கொள்வதற்கு உதவும் வழிமுறைகளை இப்பகுதியில் காணலாம். (Slicing – Dicing Methods) . மேலும் ஒரு டேட்டாஃப்பிரேம் திரையில் வெளிப்படும்போது அது எவ்வாறு அமைய வேண்டும் என்பதைக் கட்டுப்படுத்துவதற்கு உதவும் ஒருசில பண்புகளையும் இப்பகுதியில் காணலாம்.

```
import pandas as pd

df = pd.read_csv("./girls.csv")

print (df.shape)

print (df.columns)

print (df['fname']) # df.fname , df['fname','age']



print (df.loc[ 1:4 , 'fname' ])

print (df.loc[ [1,4] , 'fname' ])

print (df.loc[ : , 'age']>30 )
```

```
print (df.ix[1:4, 'fname' ])

print (df.ix[[1,4], 'fname' ])

print (df.ix[ : , 'age']>30 )



print (df.iloc[1:4,1:4])

print (df.iloc[[1,4],[1,4]])

print (df.iloc[[1,4],lambda x : [1,4]])



print (pd.get_option("display.max_rows")) # max_columns



pd.set_option("display.max_rows",5) # reset_option()

pd.set_option("display.max_columns",4) # reset_option()

print (df)



pd.reset_option("display.max_rows")

pd.reset_option("display.max_columns")

print (pd.describe_option("display.max_rows"))

with pd.option_context("display.max_rows",5):

   print (df)
```

Code link view raw 06_pandas_locdisp.py

இங்கு girls.csv எனும் கோப்பில் ஒருசில யுவதிகளின் பெயர் , துணைப் பெயர், அவர்களின் வயது, அவர்கள் வகிக்கும் பதவி, அவர்களுடைய தொலைபேசி எண் மற்றும் அவர்கள் வசிக்கும் ஊர் ஆகிய விவரங்கள் எல்லாம் கொடுக்கப்பட்டுள்ளன . இவற்றையெல்லாம் ஒரு டேட்டாஃப்பிரேமாக மாற்றிய பின்னர் அதன்மீது செயல்படும் shape எனும் பண்பு அதிலுள்ள rows, columns- ஐ வெளிப்படுத்தும். அதாவது எத்தனை யுவதிகளின் விவரங்கள் எத்தனை தலைப்புகளின் கீழ் கொடுக்கப்பட்டுள்ளன எனும் எண்ணை வெளிக்காட்டும். columns எனும் பண்பு என்னென்ன தலைப்புகளின் கீழ் கொடுக்கப்பட்டுள்ளன என்பதை வெளிக்காட்டும்.

```
df = pd.read_csv(“./girls.csv")
print (df.shape)
(10, 7)
print (df.columns)
Index(['id', 'fname', 'lname', 'age', 'desig', 'no', 'place'],
dtype='object')
```

ஏதேனும் ஒரு column இன் கீழ் அமையும் விவரங்களைப் பெற df.fname எனக் கொடுக்கலாம் அல்லது df['fname'] எனவும் கொடுக்கலாம். ஒன்றுக்கும் மேற்பட்ட columns- ஐக் கூட இம்முறையிலேயே கொடுக்கலாம்.

```
print (df['fname']) # df.fname , df['fname','age']
0 Nithya
1 Nandhini
2 Madhuri
3 Kavitha
4 Vijaya
5 Aarthi
6 Lavanya
7 Meena
8 Gayathri
9 Kavitha
Name: fname, dtype: object
```

ஏதேனும் ஒரு குறிப்பிட்ட column-ன் கீழ் , ஒரு குறிப்பிட்ட row- இல் இருக்கும் தகவல்களை மட்டும் பெற loc எனும் பண்பு பயன்படுகிறது.
இங்கு 1 முதல் 4 முடிய index எண்ணைக் கொண்ட தரவுகள் மட்டும் எடுத்துக் காட்டப்பட்டுள்ளன .

```
print (df.loc[ 1:4 , 'fname' ])
1 Nandhini
2 Madhuri
3 Kavitha
4 Vijaya
Name: fname, dtype: object
```

இங்கு 1 மற்றும் 4 ஆகிய index எண்ணைக் கொண்ட தரவுகள் மட்டும் எடுக்கப்பட்டுள்ளன.

```
print (df.loc[ [1,4] , 'fname' ])
1 Nandhini
4 Vijaya
Name: fname, dtype: object
```

இங்கு age எனும் column-ல் இருந்து அனைத்து row-ம் எடுக்கப்பட்டுள்ளன. ஆனால் relational operator மூலம் ஒப்பிட்டு 30 வயதுக்கு மேல் மதிப்பு கொண்டதை True எனவும், குறைவாக உள்ளதை False எனவும் வெளிப்படுத்தியுள்ளது.

```
print (df.loc[ : , 'age']>30 )
0 True
1 False
2 True
3 True
4 True
5 False
6 False
7 True
8 True
9 True
Name: age, dtype: bool
```

loc மற்றும் iloc ஆகிய இரண்டின் பண்புகளையும் ஒருங்கே பெற்றது ix ஆகும். இதுவும் loc பண்பு செய்யும் அதே வேலையைத்தான் செய்கிறது. இவை பின்வருமாறு.

```
print (df.ix[1:4, 'fname' ])
1 Nandhini
2 Madhuri
3 Kavitha
4 Vijaya
Name: fname, dtype: object
```

```
print (df.ix[[1,4], 'fname' ])
1 Nandhini
4 Vijaya
Name: fname, dtype: object


print (df.ix[ : , 'age']>30 )
0 True
1 False
2 True
3 True
4 True
5 False
6 False
7 True
8 True
9 True
Name: age, dtype: bool
```

iloc என்பதில் labels போன்று எதையும் கொடுத்து பிரித்தெடுக்க முடியாது. label- க்கான பெயர்கள் கொடுக்க வேண்டிய இடத்திலும் அதற்கான index மதிப்புகளையே கொடுக்க வேண்டும். இவை பின்வருமாறு.

இங்கு column-க்கான பெயர்கள் கொடுக்க வேண்டிய இடத்தில் 1:4 எனும் range கொடுக்கப்பட்டுள்ளதை கவனிக்கவும். upper லிமிட் 4-ஐ விடுத்து அதற்கு முந்தைய எண் கொண்ட column வரை கணக்கில் எடுத்துக்கொண்டது.

```
print (df.iloc[1:4,1:4])
  fname    lname     age
1 Nandhini Babu      28
2 Madhuri  Nathan    51
3 Kavitha  Manoharan 45
```

இங்கு 1,4 ஆகிய column-ல் இருக்கும் தரவுகள் மட்டும் கொடுக்கப்பட்ட row index-க்கு ஏற்ப எடுத்துக் காட்டப்பட்டுள்ளது .

```
print (df.iloc[[1,4],[1,4]])
  fname    desig
1 Nandhini Assistant Manager
4 Vijaya   AVP
```

அதே செயல் lambda function மூலம் செய்யப்பட்டுள்ளது.

```
print (df.iloc[[1,4],lambda x : [1,4]])
  fname    desig
1 Nandhini Assistant Manager
4 Vijaya   AVP
```

ஒரு டேட்டாஃப்ப்பிரேமில் 700 rows மற்றும் 250 columns இருக்கிறதெனில், அதனை பிரிண்ட் செய்யும்போது அவை அனைத்தையும் திரையில் வெளிப்படுத்தாது. அதிகபட்சம் இவ்வளவு rows மற்றும் columns –ஐத்தான் வெளிப்படுத்த வேண்டும் என்பதற்கு வரையறை உண்டு. அதனைத் தெரிந்து கொள்ள get_option என்பது பயன்படும்.இங்கு அதிகபட்சம் 60 rows வரை வெளிப்படலாம் என்று வெளிப்படுத்தியுள்ளது.

```
print (pd.get_option(“display.max_rows")) # max_columns
60
```

இம்மதிப்பினை மாற்றி அமைக்க set_option() என்பது பயன்படும். இதன் மூலம் அதிகபட்சம் 5 rows மற்றும் 4 columns வெளிப்படுமாறு செய்துள்ளோம். ஆகவேதான் டேட்டாஃப்ப்பிரேமை பிரிண்ட் செய்யும்போது, கொடுக்கப்பட்டுள்ள எண்ணிக்கைக்கு ஏற்றவாறு தரவுகளை வெளிப்படுத்தி விட்டு இடையில் இருப்பவற்றை … எனக் காட்டி உள்ளது .

```
pd.set_option(“display.max_rows",5) # reset_option()
pd.set_option(“display.max_columns",4) # reset_option()
print (df)
   id fname    … no         place
0  1  Nithya   … 9587412536 Hyderabad
1  2  Nandhini … 9848022338 Delhi
.. .. …        … …          …
8  9  Gayathri … 9848022333 Chennai
9  10 Kavitha  … 9848022336 Noida
[10 rows x 7 columns]
```

reset_option என்பது மீண்டும் அதன் default மதிப்பினைப் பெற உதவும்.

```
pd.reset_option(“display.max_rows")
pd.reset_option(“display.max_columns")
```

இது பற்றிய கூடுதல் விவரங்களைப் பெற describe_option பயன்படும்.

```
print (pd.describe_option("display.max_rows"))
display.max_rows : int
If max_rows is exceeded, switch to truncate view. Depending on
`large_repr`, objects are either centrally truncated or printed as
a summary view. 'None' value means unlimited.
In case python/IPython is running in a terminal and `large_repr`
equals 'truncate' this can be set to 0 and pandas will auto-detect
the height of the terminal and print a truncated object which fits
the screen height. The IPython notebook, IPython qtconsole, or
IDLE do not run in a terminal and hence it is not possible to do
correct auto-detection.
[default: 60] [currently: 60]
None
```

[illegible] rows,columns [illegible], with statement-[illegible] [illegible] option_context [illegible].

```
with pd.option_context("display.max_rows",5):
   print (df)
   id fname    lname      age  desig      no         place
0  1  Nithya   Duraisamy 31   Manager    9587412536 Hyderabad
1  2  Nandhini Babu      28   AstManager 9848022338 Delhi
.. .. …       …        …  …        …         …
8  9  Gayathri Ragu      36   Engineer   9848022333 Chennai
9  10 Kavitha  Manoharan 49   AVP        9848022336 Noida
[10 rows x 7 columns]
```

7 [illegible] [illegible] Pandas-7

7.1 SQL Operations

SQL- [illegible] [illegible] limit, group by, order by, joins, where condition [illegible] [illegible] [illegible] [illegible] [illegible] pandas-[illegible] [illegible]. [illegible] [illegible] [illegible] [illegible].

```
import pandas as pd

df = pd.read_csv("./girls.csv")

print (df)



print (df.head())

print (df.head(2))

print (df[df['place'] == 'Hyderabad'])



df1 = df.groupby('place')

print (df1.size())



print (df1.groups)

print (df1.get_group('Delhi'))

print (df1.filter(lambda x: len(x) >= 3))
```

```
for i, j in df1:
    print (I)
    print (j)


df2 = pd.read_csv("./boys.csv")


print (df2)


print (pd.concat([df,df2]))


print (df.append(df2))


print (pd.merge(df,df2,on='place'))


print (pd.merge(df,df2,on='place',how='right')) # left, outer, inner


print (df2.sort_index(ascending=False))


print (df2.sort_values(by='age'))
```

Code link view raw 07_pandas_sql.py

□□□□ □□□□□□□□□□□□ □□□ csv- □ □□□□□□□ □□□□□□□□□□□□□ □□□□□□□□.

```
df = pd.read_csv(“./girls.csv")
print (df)


  id fname    lname     age desig      no         place
0 1  Nithya   Duraisamy 31  Manager    9587412536 Hyderabad
1 2  Nandhini Babu      28  AstManager 9848022338 Delhi
2 3  Madhuri  Nathan    51  VP         9848022339 Delhi
3 4  Kavitha  Manoharan 45  AVP        9848022330 Hyderabad
4 5  Vijaya   Kandasamy 40  AVP        9848022336 Noida
5 6  Aarthi   Raj       22  AstManager 9848022335 Chennai
6 7  Lavanya  Sankar    23  SrEngineer 9848022334 Chennai
7 8  Meena    Baskar    56  VP         9848022333 Hyderabad
8 9  Gayathri Ragu      36  Engineer   9848022333 Chennai
9 10 Kavitha  Manoharan 49  AVP        9848022336 Noida
```

7.1.1 Limit clause

அதனை பிரிண்ட் செய்யும்போது ஒருசில ரெகார்ட்ஸை மட்டும் வெளிப்படுத்த விரும்பினால் head() – ஐப் பயன்படுத்தலாம்.

```
print (df.head())

  id fname    lname     age desig      no         place
0 1  Nithya   Duraisamy 31  Manager    9587412536 Hyderabad
1 2  Nandhini Babu      28  AstManager 9848022338 Delhi
2 3  Madhuri  Nathan    51  VP         9848022339 Delhi
3 4  Kavitha  Manoharan 45  AVP        9848022330 Hyderabad
4 5  Vijaya   Kandasamy 40  AVP        9848022336 Noida

print (df.head(2))

  id fname    lname     age desig      no         place
0 1  Nithya   Duraisamy 31  Manager    9587412536 Hyderabad
1 2  Nandhini Babu      28  AstManager 9848022338 Delhi
```

7.1.2 Where condition

ஒரு condition- ஐக் கொடுத்து அதில் பொருந்தும் தரவுகளை மட்டும் எடுக்க விரும்பினால் அது பின்வருமாறு அமையும். இங்கு ஹைதராபாத்தில் இருக்கும் நபர்களின் விவரங்கள் மட்டும் எடுத்துக் காட்டப்பட்டுள்ளது .

```
print (df[df['place'] == 'Hyderabad'])

  id fname    lname     age desig      no         place
0 1  Nithya   Duraisamy 31  Manager    9587412536 Hyderabad
3 4  Kavitha  Manoharan 45  AVP        9848022330 Hyderabad
7 8  Meena    Baskar    56  VP         9848022333 Hyderabad
```

7.1.3 Grouping

நகர வாரியாக நபர்களை குழுக்களாகப் பிரிக்க groupby () பயன்படுகிறது. அவ்வாறு பிரித்து df1 எனும் பெயரில் சேமிக்கிறது. அதன் மீது செயல்படும் size () என்பது என்னென்ன குழுக்களில் எவ்வளவு records உள்ளது என்பதை வெளிப்படுத்தி உள்ளது.

```
df1 = df.groupby('place')
print (df1.size())


place
Chennai   3
Delhi     2
Hyderabad 3
Noida     2
dtype: int64
```

df1 –□□ □□□□ □□□□□□□□□□ groups □□□□□□ □□□□□□□□□□□ □□□□□, □□□□□ □□□□□□□□□□□□□□□□ row-□□ □□□□□□□□□ □□□, □□□□□□□□ □□□□ □□□ □□□□□□□□□ □□□□□□□□□□□□□□□.

```
print (df1.groups)


{'Chennai': Int64Index([5, 6, 8], dtype='int64′), 'Delhi':
Int64Index([1, 2], dtype='int64′), 'Hyderabad': Int64
Index([0, 3, 7], dtype='int64′), 'Noida': Int64Index([4, 9],
dtype='int64′)}
```

□□□□□□□ □□□ □□□□□□□□ □□□□□□□ □□□□□□□□ □□□□□ □□□□□□□□□□□□□□□□□ □□□□□□□□□ □□□□□□□ □□□□□□ □□□□□□□□□□□□ get_group() □□□□□□□□□□.

```
print (df1.get_group('Delhi'))


  id fname    lname     age desig      no         place
1 2  Nandhini Babu      28  AstManager 9848022338 Delhi
2 3  Madhuri  Nathan    51  VP         9848022339 Delhi
```

□□□□ □□□□□□□□ □□□□□□ □□□□□□ □□□□□□□□ □□□□□□□□ □□□□□□□□ □□□□□□, □□□ □□□□□□□ filter □□□□□□ □□□□□□□ □□□□□□□□□□□□ □□□ □□□□□□□□□□□□.

```
print (df1.filter(lambda x: len(x) >= 3))


  id fname    lname     age desig      no         place
0 1  Nithya   Duraisamy 31  Manager    9587412536 Hyderabad
3 4  Kavitha  Manoharan 45  AVP        9848022330 Hyderabad
5 6  Aarthi   Raj       22  AstManager 9848022335 Chennai
6 7  Lavanya  Sankar    23  SrEngineer 9848022334 Chennai
```

```
7 8  Meena    Baskar    56  VP          9848022333 Hyderabad
8 9  Gayathri Ragu      36  Engineer    9848022333 Chennai
```

□□□□□□□□□ □□□□□□□□□□□□ □□□□□□□ □□□□ (df1), □□□□□□□□ □□□□□, □□□□□ □□□□□□□□□□□□□□□□□□ □□□□□□□□□ □□□□□□□□□ □□□□□□□ □□□ □□□□□□□□□□□□ □□□□□□□□□□□ □□□□□□□ for loop □□□□□□□□□□□ □□□□□□□.

```
for i, j in df1:
    print (i)
    print (j)


Chennai
  id fname    lname     age desig      no         place
5 6  Aarthi   Raj       22  AstManager 9848022335 Chennai
6 7  Lavanya  Sankar    23  SrEngineer 9848022334 Chennai
8 9  Gayathri Ragu      36  Engineer   9848022333 Chennai



Delhi
  id fname    lname     age desig      no         place
1 2  Nandhini Babu      28  AstManager 9848022338 Delhi
2 3  Madhuri  Nathan    51  VP         9848022339 Delhi



Hyderabad
  id fname    lname     age desig      no         place
0 1  Nithya   Duraisamy 31  Manager    9587412536 Hyderabad
3 4  Kavitha  Manoharan 45  AVP        9848022330 Hyderabad
7 8  Meena    Baskar    56  VP         9848022333 Hyderabad



Noida
  id fname    lname     age desig      no         place
4 5  Vijaya   Kandasamy 40  AVP        9848022336 Noida
9 10 Kavitha  Manoharan 49  AVP        9848022336 Noida
```

7.1.4 Combining data frames

□□□□□□□□□ □□□□□□□ □□□□□□□□□□□□□□□ □□□□□□□□□, □□□□□□□□□□ □□□□□□□□□ □□□□□□□□□□□□□□□□ □□□□ □□□□□□□□□□ □□□□□□□□□□□□ □□□□□□□.

```
df2 = pd.read_csv(“./boys.csv")
print (df2)


  id fname     lname    age desig     no         place
0 11 Mahesh    Muthu    38  SrManager 9853526341 Chennai
1 12 Elango    Raman    25  Assistant 9865637872 Mumbai
2 13 Kadiresan kesavan  41  VP        9746325437 Delhi
3 14 Sundar    Paul     48  AVP       9947362222 Thane
4 15 Kumar     Ramasamy 30  AVP       9346212333 Noida


print (pd.concat([df,df2]))


  id fname     lname     age desig      no         place
0 1  Nithya    Duraisamy 31 Manager     9587412536 Hyderabad
1 2  Nandhini  Babu      28 AstManager  9848022338 Delhi
2 3  Madhuri   Nathan    51 VP          9848022339 Delhi
3 4  Kavitha   Manoharan 45 AVP         9848022330 Hyderabad
4 5  Vijaya    Kandasamy 40 AVP         9848022336 Noida
5 6  Aarthi    Raj       22 AstManager  9848022335 Chennai
6 7  Lavanya   Sankar    23 SrEngineer  9848022334 Chennai
7 8  Meena     Baskar    56 VP          9848022333 Hyderabad
8 9  Gayathri  Ragu      36 Engineer    9848022333 Chennai
9 10 Kavitha   Manoharan 49 AVP         9848022336 Noida
0 11 Mahesh    Muthu     38 SrManager   9853526340 Chennai
1 12 Elango    Raman     25 Assistant   9865637872 Mumbai
2 13 Kadiresan kesavan   41 VP          9746325437 Delhi
3 14 Sundar    Paul      48 AVP         9947362222 Thane
4 15 Kumar     Ramasamy  30 AVP         9346212333 Noida


print (df.append(df2))


  id fname     lname     age desig      no         place
0 1  Nithya    Duraisamy 31 Manager     9587412536 Hyderabad
1 2  Nandhini  Babu      28 AstManager  9848022338 Delhi
2 3  Madhuri   Nathan    51 VP          9848022339 Delhi
3 4  Kavitha   Manoharan 45 AVP         9848022330 Hyderabad
4 5  Vijaya    Kandasamy 40 AVP         9848022336 Noida
5 6  Aarthi    Raj       22 AstManager  9848022335 Chennai
6 7  Lavanya   Sankar    23 SrEngineer  9848022334 Chennai
7 8  Meena     Baskar    56 VP          9848022333 Hyderabad
8 9  Gayathri  Ragu      36 Engineer    9848022333 Chennai
9 10 Kavitha   Manoharan 49 AVP         9848022336 Noida
0 11 Mahesh    Muthu     38 SrManager   9853526340 Chennai
1 12 Elango    Raman     25 Assistant   9865637872 Mumbai
```

```
2 13 Kadiresan kesavan   41 VP         9746325437 Delhi
3 14 Sundar    Paul      48 AVP        9947362222 Thane
4 15 Kumar     Ramasamy  30 AVP        9346212333 Noida
```

7.1.5 Joins

[illegible] [illegible], [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible], [illegible] join [illegible] [illegible].

```
print (pd.merge(df,df2,on='place'))


  id_x fname_x  lname_x   age_x … lname_y  age_y desig_y   no_y
0 2    Nandhini Babu      28    … kesavan  41    VP        9746325437
1 3    Madhuri  Nathan    51    … kesavan  41    VP        9746325437
2 5    Vijaya   Kandasamy 40    … Ramasamy 30    AVP       9346212333
3 10   Kavitha  Manoharan 49    … Ramasamy 30    AVP       9346212333
4 6    Aarthi   Raj       22    … Muthu    38    SrManager 9853526340
5 7    Lavanya  Sankar    23    … Muthu    38    SrManager 9853526341
6 9    Gayathri Ragu      36    … Muthu    38    SrManager 9853526341
[7 rows x 13 columns]
```

[illegible] [illegible] left, right,inner, outer [illegible] [illegible] join [illegible] [illegible] [illegible].

```
print (pd.merge(df,df2,on='place',how='right')) # left, outer, inner


  id_x fname_x  lname_x   age_x… lname_y  age_y desig_y   no_y
0 2.0  Nandhini Babu      28.0 … kesavan  41    VP        9746325437
1 3.0  Madhuri  Nathan    51.0 … kesavan  41    VP        9746325437
2 5.0  Vijaya   Kandasamy 40.0 … Ramasamy 30    AVP       9346212333
3 10.0 Kavitha  Mnoharan  49.0 … Ramasamy 30    AVP       9346212333
4 6.0  Aarthi   Raj       22.0 … Muthu    38    SrManager 9853526340
5 7.0  Lavanya  Sankar    23.0 … Muthu    38    SrManager 9853526341
6 9.0  Gayathri Ragu      36.0 … Muthu    38    SrManager 9853526341
7 NaN  NaN      NaN       NaN  … Raman    25    Assistant 9865637872
8 NaN  NaN      NaN       NaN  … Paul     48    AVP       9947362222
[9 rows x 13 columns]
```

7.1.6 Sorting

□□□□□□□ □□□□□□□□□□□□□□□□□ □□□□□□ □□□□□□ □□□□□□□□□□ □□□□□□□. Index- □ □□□□□□ □□□□□□□□□□□□□□□□ □□□□□□ □□□□□□□ □□□ □□□□□□□□□□□ □□□□□ column-□□ □□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□.

□□□□□ sort_index() □□□□□ □□□□□□□ □□□□□□ □□□□□□□□□ □□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□.

□□□□□□□ sort_values() □□□□□ age column-□□ □□□□ □□□□□□□□□□□ □□□□□□□□ □□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□.

```
print (df2.sort_index(ascending=False))

  id fname     lname    age desig       no         place
4 15 Kumar     Ramasamy 30  AVP         9346212333 Noida
3 14 Sundar    Paul     48  AVP         9947362222 Thane
2 13 Kadiresan kesavan  41  VP          9746325437 Delhi
1 12 Elango    Raman    25  Assistant   9865637872 Mumbai
0 11 Mahesh    Muthu    38  Sr Manager  9853526340 Chennai

print (df2.sort_values(by='age'))

  id fname     lname    age desig      no place
1 12 Elango    Raman    25  Assistant 9865637872 Mumbai
4 15 Kumar     Ramasamy 30  AVP        9346212333 Noida
0 11 Mahesh    Muthu    38  SrManager 9853526340 Chennai
2 13 Kadiresan kesavan  41  VP         9746325437 Delhi
3 14 Sundar    Paul     48  AVP        9947362222 Thane
```

8 எளிய தமிழில் Pandas-8

8.1 Loops & Functions

ஒரு டேட்டாஃப்ப்பிரேமில் உள்ளவற்றை for லூப் மூலம் பல்வேறு விதங்களில் வெளிப்படுத்திக் காட்டலாம். அவை கீழே கொடுக்கப்பட்டுள்ளன. அடுத்து lambda எனும் ஒற்றை வரி பங்ஷன் மூலமும், user defined function மூலமும் டேட்டாஃப்ப்பிரேம் மதிப்புகளில் மாற்றம் செய்வது எப்படி என்று காட்டப்பட்டுள்ளது . கடைசியாக reindex_like() எனும் பங்ஷன் மூலம் ஒரு டேட்டாஃப்ப்பிரேமின் வடிவத்தை மற்றொரு டேட்டாஃப்ப்பிரேமைப் போலவே அமைப்பது எப்படி என்று காட்டப்பட்டுள்ளது .

```
import pandas as pd

import numpy as np

l2 = [[90,83,67,83,45],[68,89,75,56,73],[58,88,60,90,100]]

df =
pd.DataFrame(l2,columns=['Tamil','English','Maths','Science','S
ocial'],index=['Ramesh','Suresh','Kamesh'],dtype='int32')

for i in df:
    print (i)

for i in df.itertuples():
    print (i)

for i,j in df.iteritems():
    print (I)
    print (j)

for i,j in df.iterrows():
    print (I)
    print (j)

print (df['Tamil'].isin ([68,58]))
print (df.apply(lambda x: x+2))
```

```
def hi(a,b):
   return a+b

print (df.pipe(hi,2))

l2 = [[90,83,45],[68,89,73],[58,88,100]]

df2 = 
pd.DataFrame(l2,columns=['Tamil','English','Social'],index=['Ra
mesh','Suresh','Jagadesh'],dtype='int32')

print (df2)

print (df2.reindex_like(df))
```

Code link 08_pandas_loopsfunc.py

□□□□□□□□□□□□□□□□ □□□□ □□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□ □□□□□□□□□□□.

```
l2 = [[90,83,67,83,45],[68,89,75,56,73],[58,88,60,90,100]]

df = 
pd.DataFrame(l2,columns=['Tamil','English','Maths','Science','S
ocial'],index=['Ramesh','Suresh','Kamesh'],dtype='int32′)
```

□□□□□ for □□□□□ □□□□□□□ □□□□□ □□□□□□□□□□□□□□□□□□□□ iterate □□□□□□□□□□, □□□□ □□□□□□□□□□□□□□□□ □□□□□□□□□□□ □□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□.

```
for i in df:
    print (i)


Tamil
English
Maths
Science
Social
```

[illegible] itertuples [illegible], [illegible] row-[illegible].

```
for i in df.itertuples():
    print (i)


Pandas(Index='Ramesh', Tamil=90, English=83, Maths=67, Science=83,
Social=45)
Pandas(Index='Suresh', Tamil=68, English=89, Maths=75, Science=56,
Social=73)
Pandas(Index='Kamesh', Tamil=58, English=88, Maths=60, Science=90,
Social=100)
```

[illegible] iteritems [illegible] iterate [illegible], column-[illegible] row wise [illegible].

```
for i,j in df.iteritems():
    print (i)
    print (j)


Tamil
Ramesh 90
Suresh 68
Kamesh 58
Name: Tamil, dtype: int32
.
.
.
.
Social
Ramesh 45
Suresh 73
Kamesh 100
Name: Social, dtype: int32
```

iterrows [illegible] row-[illegible] column wise [illegible].

```
for i,j in df.iterrows():
    print (i)
    print (j)
```

```
Ramesh
Tamil 90
English 83
Maths 67
Science 83
Social 45
Name: Ramesh, dtype: int32
.
.
.
Kamesh
Tamil 58
English 88
Maths 60
Science 90
Social 100
Name: Kamesh, dtype: int32
```

ஒரு டேட்டாஃப்ப்பிரேம் column-ல் உள்ள மதிப்புகள் குறிப்பிட்ட சில மதிப்புகளைக் கொண்டிருக்கின்றனவா இல்லையா என்பதை சோதிக்க isin என்பது பயன்படுகிறது.

```
print (df['Tamil'].isin ([68,58]))


Ramesh False
Suresh True
Kamesh True
Name: Tamil, dtype: bool
```

Lambda எனும் ஒற்றை வரி பங்ஷன் மூலம் அனைத்து மதிப்புகளுடனும் 2 கூட்டி காட்டப்பட்டுள்ளது .

```
print (df.apply(lambda x: x+2))


       Tamil English Maths Science Social
Ramesh 92    85      69    85      47
Suresh 70    91      77    58      75
Kamesh 60    90      62    92      102
```

அதே வேலை hi எனும் user defined function கொண்டு செய்யப்பட்டுள்ளது. டேட்டாஃப்ப்பிரேமுடன் அந்த பங்ஷனை இணைக்க pipe பயன்பட்டுள்ளது.

```
def hi(a,b):
    return a+b
print (df.pipe(hi,2))
```

இங்கு df2 எனும் இரண்டாவது டேட்டாஃப்ப்பிரேம் உருவாக்கப்பட்டுள்ளது. இதனை df-வுடன் ஒப்பிடும்போது ரமேஷ், சுரேஷ் ஆகிய இரண்டு இன்டெக்ஸ் மற்றும் தமிழ், இங்கிலீஷ், சோஷியல் ஆகிய மூன்று columns-ஐ மட்டும் பொதுவாகப் பெற்றுள்ளது.

```
l2 = [[90,83,45],[68,89,73],[58,88,100]]
df2 =
pd.DataFrame(l2,columns=['Tamil','English','Social'],index=['Ramesh',
'Suresh','Jagadesh'],dtype='int32′)
print (df2)


         Tamil English Social
Ramesh   90    83      45
Suresh   68    89      73
Jagadesh 58    88      100
```

எனவே df2-ன் இன்டெக்ஸை df போலவே அமைக்கும்போது இல்லாத rows, columns –க்கு null மதிப்பினை இட்டு வடிவத்தை அதே போலவே அமைக்கும் .

```
print (df2.reindex_like(df))


       Tamil English Maths Science Social
Ramesh 90.0  83.0    NaN   NaN     45.0
Suresh 68.0  89.0    NaN   NaN     73.0
Kamesh NaN   NaN     NaN   NaN     NaN
```

9 எளிய தமிழில் Pandas-9

9.1 Metrics

தரவுகளைப் பற்றிய புரிதலை இன்னும் நுணுக்கமாகத் துல்லியமாக அமைப்பதற்கு பல்வேறு அளவீடுகள் உதவுகின்றன. அவைகளின் பட்டியலை இப்பகுதியில் காணலாம். பொதுவாக இதுபோன்ற அளவீடுகளைக் கணக்கிடுவதற்கு இந்த உதாரணம் பொருத்தமாக இருக்காது. ஆனாலும் முதலில் இதை வைத்துப் புரிந்து கொண்டால், பின்னர் பெரிய அளவிலான தரவுகளை கையாளும் போது சுலபமாக இருக்கும் என்பதற்காக அதே உதாரணத்தை நான் பயன்படுத்தியுள்ளேன்.

```
import pandas as pd

import numpy as np

l2 = [[90,83,67,83,45],[68,89,75,56,73],[58,88,60,90,100]]

df =
pd.DataFrame(l2,columns=['Tamil','English','Maths','Science','Social'
],index=['Ramesh','Suresh','Kamesh'],dtype='int32')

print (df)

print (df.pct_change())

print (df['Tamil'].cov(df['English']))

print (df.cov())

print (df.corr())

print (df.rank())

print (df.rolling(window=3).mean())

print (df.rolling(window=3,min_periods=2).mean())
```

```
print
(df.rolling(window=3,min_periods=2).aggregate([np.sum,np.mean]))


print (df.rolling(window=2).sum())


print (df.expanding(min_periods=2).sum())


print (df.ewm(com=0.5).mean())
```

view raw 09_pandas_metrics.py

முதலில் ஒரு டேட்டாஃப்பிரேமை உருவாக்கிக் கொள்வோம்.

```
l2 = [[90,83,67,83,45],[68,89,75,56,73],[58,88,60,90,100]]
df =
pd.DataFrame(l2,columns=['Tamil','English','Maths','Science','Social'
],index=['Ramesh','Suresh','Kamesh'],dtype='int32′)
```

பின் அதனை பிரிண்ட் செய்து பார்ப்போம்.

```
print (df)


       Tamil English Maths Science Social
Ramesh 90    83      67    83      45
Suresh 68    89      75    56      73
Kamesh 58    88      60    90      100
```

9.1.1 Percentage Change

ஒவ்வொரு row-வில் உள்ள தரவுகளும் அதற்கு முந்தைய row-வில் உள்ள தரவுகளுடன் எவ்வளவு சதவீதம் மாறுபடுகின்றன என்பதைக் கண்டுபிடிக்க pct_change() எனும் ஃபங்ஷன் பயன்படுகிறது.

ஆகவேதான் முதல் row-க்கு முந்தி எந்த ஒரு row-வும் இல்லாத காரணத்தால் NaN என்பதை வெளிப்படுத்தியுள்ளது. அதற்கு அடுத்த row-வில் (68-90)/90 = -0.244444 என்பதை வெளிப்படுத்தியுள்ளது. இதே போல் ஒவ்வொரு மதிப்பையும் கணக்கிட்டு வெளிப்படுத்தியுள்ளது.

```
print (df.pct_change())
```

```
       Tamil     English   Maths     Science   Social
Ramesh NaN       NaN       NaN       NaN       NaN
Suresh -0.244444 0.072289  0.119403  -0.325301 0.622222
Kamesh -0.147059 -0.011236 -0.200000 0.607143  0.369863
```

9.1.2 Covariance

ஒவ்வொரு column-ல் உள்ள தரவுகளும் மற்ற column-ல் உள்ள தரவுகளுடன் எந்த அளவுக்குத் தொடர்புடையதாக இருக்கின்றன என்பதைக் கண்டுபிடிக்க Covariance,Correlation ஆகியவை பயன்படுகின்றன. இவற்றின் முக்கியத்துவத்தை மெஷின் லேர்னிங் என்ற புத்தகத்தில் ஏற்கனவே பார்த்தோம். இப்போது இவற்றுக்கான கணக்கீடுகளைப் பற்றி கொஞ்சம் கற்றுக் கொள்வோம்.

ஒவ்வொரு column-ல் இருக்கும் எண்களும் பல்வேறு range-ல் இருக்க வாய்ப்புகள் உள்ளன. ஆகவேதான் இவற்றுக்கிடையே இருக்கும் தொடர்பினைக் கண்டறிய ஒவ்வொரு column-ல் உள்ள மதிப்புகளுடனும் அந்த column-ன் சராசரி கழிக்கப்பட்டு ஒரு பொதுமதிப்பு உருவாக்கப்படுகிறது. இவ்வாறு இரண்டு column-க்கும் உருவாக்கப்பட்ட பொது மதிப்புகளை வைத்து அதன் தொடர்பு கண்டுபிடிக்கப்படுகிறது.
Covariance-க்கான வாய்ப்பாடு பின்வருமாறு.

```
summation of (x element – mean(x)).(y element – mean(y))/ N -1
```

இங்கு தமிழ், ஆங்கிலம் ஆகிய இரு column-ல் சேமிக்கப்பட்டுள்ள தரவுகளின் covariance மதிப்பு பின்வருமாறு கணக்கிடப்படுகிறது.

```
(90+68+58)/3 = 72
(83+89+88)/3 = 86.67


= {[(90-72)*(83-86.67)]+
[(68-72)*(89-86.67)]+
[(58-72)*(88-86.67)]}/(3-1)
= {[18*(-3.67)]+
[(-4)*2.33]+
[(-14)*1.33]}/2
= [(-66.06)+(-9.32)+(-18.62)]/2
= -94/2 = -47


print (df['Tamil'].cov(df['English']))
-47.0
```

அடுத்து ஒவ்வொரு column-க்கும் மற்ற column-களுக்கு இடையேயான covariance மதிப்பு கண்டுபிடிக்கப்பட்டுள்ளது .

```
print (df.cov())

        Tamil    English    Maths       Science     Social
Tamil   268.0    -47.000000 33.000000   5.000000    -441.000000
English -47.0    10.333333  4.666667    -26.833333  69.333333
Maths   33.0     4.666667   56.333333   -129.166667 -94.333333
Science 5.0 –    26.833333  -129.166667 322.333333  91.166667
Social  -441.0   69.333333  -94.333333  91.166667   756.333333
```

9.1.3 Correlation

Correlation-க்கான வாய்ப்பாடு பின்வருமாறு.

```
summation of (x element – mean(x)).(y element – mean(y))
/ square root of {summation of {square of [(x element –
mean(x)]. square of [(y element – mean(y)]}}
```

Covariance மதிப்பு infinity வரை செல்லும். இதன் அளவிடப்பட்ட வடிவமாக (scaled form) correlation-ஐ சொல்லலாம். correlation மதிப்பு பொதுவாக -1 லிருந்து 1 வரை அமையும்.

மேலும் correlation மதிப்பு தரவுகளில் ஏற்படும் மாறுதல்களுக்கு ஏற்ப மாறி அமையாது. எடுத்துக்காட்டாக இரு column-லும் உள்ள அனைத்து மதிப்புகளையும் ஒரு குறிப்பிட்ட எண்ணால் பெருக்குகிறோம் என்று வைத்துக் கொள்வோம் . இந்த மாறுதல்களால் covariance மதிப்பு மாறுபடும் . ஆனால் இது போன்ற மாற்றங்கள் correlation மதிப்பில் எவ்வித மாற்றத்தையும் உண்டாக்காது.

இங்கு தமிழ், ஆங்கிலம் ஆகிய இரு column-ல் சேமிக்கப்பட்டுள்ள தரவுகளின் correlation மதிப்பு பின்வருமாறு கணக்கிடப்படுகிறது.

```
(90+68+58)/3 = 72
(83+89+88)/3 = 86.67


= {[18*(-3.67)]+[(-4)*2.33]+
[(-14)*1.33]} /
sqft{[(18*18)+(-4*-4)+(-14*-14]
```

```
*[(-3.67*-3.67)+(2.33*2.33)+(1.33*1.33)]}
= [(-66.06)+(-9.32)+(-18.62)]/
sqft{[324+16+196]
*[13.468+5.4289+1.7689]}
= -94/sqft(536*20.6658)
= -94/105.2467 = -0.8931


print (df.corr())


        Tamil     English   Maths     Science   Social
Tamil   1.000000  -0.893121 0.268574  0.017012  -0.979523
English -0.893121 1.000000  0.193421  -0.464945 0.784269
Maths   0.268574  0.193421  1.000000  -0.958551 -0.457010
Science 0.017012  -0.464945 -0.958551 1.000000  0.184640
Social  -0.979523 0.784269  -0.457010 0.184640  1.000000
```

9.1.4 Ranks

கொடுக்கப்பட்ட மதிப்புகளின் அடிப்படையில் தரவரிசைப் படுத்திக் காட்ட rank() பயன்படுகிறது.

```
print (df.rank())


       Tamil English Maths Science Social
Ramesh 3.0   1.0     2.0   2.0     1.0
Suresh 2.0   3.0     3.0   1.0     2.0
Kamesh 1.0   2.0     1.0   3.0     3.0
```

9.1.5 Rolling

Rolling Mean என்றால் நகரும் சராசரி என்று பொருள். பொதுவாக timeseries தரவுகளில் இதுபோன்ற விஷயங்கள் பயன்படும். கொடுக்கப்பட்ட window மதிப்பின் அடிப்படையில் தரவுகளைப் பிரித்து அதனைக் கொண்டு சராசரியைக் கணக்கிடும். இவ்வாறு கணக்கிடும்போது ஒவ்வொரு row-ம் விண்டோ அளவிற்கு ஏற்ப அதற்கு முந்தைய ரெகார்ட்ஸை எடுத்துக்கொண்டு கணக்கிடும். ஆகவே சராசரி கணக்கெடுப்புக்கு உதவும் rows-ஆனது சிறு சிறு பகுதிகளாக நகர்ந்து கொண்டே செல்லும். எனவேதான் இது Moving Average என்றும் அழைக்கப்படுகிறது. இங்கு விண்டோ அளவு மூன்று என கொடுக்கப்பட்டுள்ளதால் முதலிரண்டு row-க்கு null மதிப்பை வெளிப்படுத்திவிட்டு, மூன்றாவது ரோவில் இருந்து மும்மூன்று row-க்கும் சராசரியைக் கணக்கிட்டு வெளிப்படுத்தியுள்ளது.

(90+68+58)/3 = 72

நான்காவது row இருந்திருந்தால் 2,3,4 ஆகிய இடங்களில் இருக்கும் மதிப்பைக் கொண்டு சராசரியைக் கணக்கிடும்.

```
print (df.rolling(window=3).mean())


       Tamil English   Maths     Science   Social
Ramesh NaN   NaN       NaN       NaN       NaN
Suresh NaN   NaN       NaN       NaN       NaN
Kamesh 72.0  86.666667 67.333333 76.333333 72.666667
```

அதுவே min periods=2 எனக் கொடுத்தால் மூன்று row இல்லாதபோது குறைந்தபட்சம் இரண்டு row இருந்தால் கூட அதை எடுத்துக்கொண்டு சராசரியைக் கணக்கிடும். அதுவும் இல்லாத பட்சத்தில் தான் NaN-ஐ வெளிப்படுத்தும்.

இங்கு முதல் row-க்கு NaN என்பதையும், இரண்டாவது row-க்கு இரண்டின் சராசரியையும், மூன்றாவது row-லிருந்து கொடுக்கப்பட்ட விண்டோ மதிப்பின்படி சராசரியைக் கணக்கிட்டும் வெளிப்படுத்தியுள்ளது.

```
(90+68)/2 = 79
(90+68+58)/3 = 72


print (df.rolling(window=3,min_periods=2).mean())


       Tamil  English   Maths     Science   Social
Ramesh NaN    NaN       NaN       NaN       NaN
Suresh 79.0   86.000000 71.000000 69.500000 59.000000
Kamesh 72.0   86.666667 67.333333 76.333333 72.666667
```

ஒன்றுக்கும் மேற்பட்ட மெட்ரிக்சை கண்டுபிடித்து வெளிப்படுத்த விரும்பினால் aggregate () என்பதற்குள் பின்வருமாறு கொடுக்கலாம் . இங்கு sum, mean ஆகிய இரண்டும் கண்டுபிடித்து வெளிப்படுத்தப்பட்டுள்ளது.

```
90+68 = 158
90+68+58 = 216


(90+68)/2 = 79
(90+68+58)/3 = 72
```

```
print
(df.rolling(window=3,min_periods=2).aggregate([np.sum,np.mean]))


       Tamil      Englis… Science   Social
       sum   mean sum   … mean      sum   mean
Ramesh NaN   NaN  NaN   … NaN       NaN   NaN
Suresh 158.0 79.0 172.0 … 69.500000 118.0 59.000000
Kamesh 216.0 72.0 260.0 … 76.333333 218.0 72.666667
[3 rows x 10 columns]
```

9.1.6 Expanding

Rolling என்பது சிறு சிறு பகுதிகளாக தரவுகளைப் பிரித்துக்கொண்டு கணக்கீடுகளை நிகழ்த்தும். Expanding என்பது கணக்கிட்டு வந்த மதிப்புகளைக் கணக்கில் கொண்டு தனது கணக்கீடுகளைத் தொடர்ந்து கொண்டே செல்லும் .

கீழ்கண்ட Rolling எடுத்துக்காட்டில் மூன்றாவது row-க்கான கணக்கீடு அதன் மதிப்பையும் அதற்கு முந்தைய row மதிப்பான 68-ஐயும் இணைத்து 126 என்பதனை வெளிப்படுத்தியுள்ளது.

```
90+68+58


print (df.rolling(window=2).sum())


       Tamil English Maths Science Social
Ramesh NaN   NaN     NaN   NaN     NaN
Suresh 158.0 172.0   142.0 139.0   118.0
Kamesh 126.0 177.0   135.0 146.0   173.0
```

அதுவே expanding என்று வரும்போது மூன்றாவது row-க்கான கணக்கீடு அதன் மதிப்பையும் அதற்கு முந்தைய row கணக்கிட்டு வெளிப்படுத்திய மதிப்பான 158-ஐயும் இணைத்து 216 என வெளிப்படுத்தியுள்ளதைக் காணலாம்.

```
print (df.expanding(min_periods=2).sum())


       Tamil English Maths Science Social
Ramesh NaN   NaN     NaN   NaN     NaN
Suresh 158.0 172.0   142.0 139.0   118.0
Kamesh 216.0 260.0   202.0 229.0   218.0
```

9.1.7 Exponential weighted functions

இதுவரை நாம் பார்த்த rolling, expanding ஆகிய அனைத்தும் பல்வேறு யுக்திகளைக் கொண்டு mean, median ஆகியவற்றைக் கணக்கிடுகின்றன. ஆனால் இதுவோ ஒருசில parameters – ஐக் கணக்கில் கொண்டு அதே விசயத்தை செய்கின்றது .

com, span, halflife போன்ற சில parameters- ஐ வைத்து முதலில் ஆல்ஃபா மதிப்பைக் கண்டுபிடிக்கிறது.

```
alpha = 1/(1+com)
```

பின் அந்த ஆல்ஃபா மதிப்பை வைத்து கீழ்க்கண்ட சமன்பாட்டில் பொருத்தி வேண்டிய மதிப்புகள் கண்டுபிடிக்கப்படுகின்றன.

```
weighted_average[0] = arg[0]
weighted_average[i] = (1-alpha)*wt._avg[i-1]+alpha*arg[i]
```

இங்கு com = 0.5 என வைத்து ஆல்ஃபா மதிப்பு கண்டுபிடிக்கப்பட்டுள்ளது.

```
alpha = 1/(1+0.5) = 1/1.5 = 0.66
```

பின் முதல் row-க்கான mean அதே மதிப்பாக அமைந்துள்ளது.

அடுத்த row-க்கான mean சமன்பாட்டின் படி கீழ்கண்டவாறு கண்டுபிடிக்கப்பட்டுள்ளது .

```
(1-0.66)*90 + (0.66*68)
= 30.6 + 44.88 = 75.48
```

இங்கு 75.48 என கணக்கிடப்பட்டுள்ளது. ஆனால் நமது நிரல் 73.50 என்பதை வெளிப்படுத்தியுள்ளது.

மூன்றாவது row-க்கான mean கீழ்கண்டவாறு.

```
(1-0.66)*75 + (0.66*58)
= 25.5 + 38.28 = 63.78
```

இங்கு 63.78 என கணக்கிடப்பட்டுள்ளது. ஆனால் நமது நிரல் 62.76 என்பதை வெளிப்படுத்தியுள்ளது.

இந்தச் சிறு வித்தியாசம் எதனால் என்பது இன்னும் சரிவர புலப்படவில்லை . இதைப் படிக்கும் யாரேனும் இதற்கான விடையைக் கண்டுபிடித்தால் கூறவும்.

```
print (df.ewm(com=0.5).mean())


       Tamil     English   Maths Science   Social
Ramesh 90.000000 83.000000 67.0  83.000000 45.000000
Suresh 73.500000 87.500000 73.0  62.750000 66.000000
Kamesh 62.769231 87.846154 64.0  81.615385 89.538462
```

10 [illegible] Pandas-10

10.1 Handling Null values

[illegible] Null [illegible]. [illegible].

```
import pandas as pd

import numpy as np

l2 = [[90,83,67,83,45],[68,89,75,56,73],[58,88,60,90,100]]

df =
pd.DataFrame(l2,columns=['Tamil','English','Maths','Science','Social'
],index=['Ramesh','Suresh','Kamesh'],dtype='int32')

df = df.rolling(window=3).mean()

print (df)

print (df.isnull())

print (df.notnull())

print (df.dropna())

print (df.fillna(0))

print (df.fillna(method='bfill')) # pad / fill for forward fill;
#bfill / backfill from backward

print (df.fillna(method='bfill',limit=1))

print (df.sum())
```

```
print (df.replace({72:100}))
```

Code link 10_pandas_null.py

[illegible] Null [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

```
df = df.rolling(window=3).mean()
print (df)


       Tamil English   Maths     Science   Social
Ramesh NaN   NaN       NaN       NaN       NaN
Suresh NaN   NaN       NaN       NaN       NaN
Kamesh 72.0  86.666667 67.333333 76.333333 72.666667
```

[illegible] [illegible] Null [illegible] [illegible] [illegible] [illegible] isnull () [illegible]. [illegible] Null [illegible] [illegible] [illegible] [illegible] True [illegible] [illegible].

```
print (df.isnull())


       Tamil English Maths Science Social
Ramesh True  True    True  True    True
Suresh True  True    True  True    True
Kamesh False False   False False   False
```

Null [illegible] [illegible] [illegible] [illegible] notnull () [illegible]. [illegible] Null [illegible] [illegible] [illegible] True [illegible] [illegible].

```
print (df.notnull())


       Tamil English Maths Science Social
Ramesh False False   False False   False
Suresh False False   False False   False
Kamesh True  True    True  True    True
```

Null [illegible] [illegible] [illegible] [illegible] dropna () [illegible].

```
print (df.dropna())
```

```
       Tamil English   Maths     Science   Social
Kamesh 72.0  86.666667 67.333333 76.333333 72.666667
```

Null மதிப்புகளை சுழியம் மதிப்பால் இடமாற்றம் செய்ய விரும்பினால் fillna – ஐப் பயன்படுத்தலாம்.

```
print (df.fillna(0))


       Tamil English   Maths     Science   Social
Ramesh 0.0   0.000000  0.000000  0.000000  0.000000
Suresh 0.0   0.000000  0.000000  0.000000  0.000000
Kamesh 72.0  86.666667 67.333333 76.333333 72.666667
```

Null மதிப்புகளை அதற்கு அடுத்துள்ள row மதிப்புகளால் நிரப்ப விரும்பினால் method=bfill எனக் கொடுக்கலாம். இது backward fill எனப் பொருள்படும். அதேபோல வெறும் fill என்பது ஃபார்வேர்ட் fill- ஐக் குறிக்கும் .

```
print (df.fillna(method='bfill')) # pad / fill for #forward fill;
bfill / backfill from backward


       Tamil English   Maths     Science   Social
Ramesh 72.0  86.666667 67.333333 76.333333 72.666667
Suresh 72.0  86.666667 67.333333 76.333333 72.666667
Kamesh 72.0  86.666667 67.333333 76.333333 72.666667
```

method=bfill என்பதுடன் limit=1 எனக் கொடுத்தால் கீழிருந்து மேலாக ஒரே ஒரு row-ஐ மட்டும் நிரப்பும் .

```
print (df.fillna(method='bfill',limit=1))


       Tamil English   Maths     Science   Social
Ramesh NaN   NaN       NaN       NaN       NaN
Suresh 72.0  86.666667 67.333333 76.333333 72.666667
Kamesh 72.0  86.666667 67.333333 76.333333 72.666667
```

Null மதிப்புகளைக் கொண்ட டேட்டாஃப்பிரேமின் மீது sum() என்பது போன்ற aggregate functions-ஐப் பயன்படுத்தினால், அவை Null மதிப்புகளைத் தவிர்த்து மீதமுள்ள மதிப்புகளை மட்டும் கணக்கில் எடுத்துக் கொண்டு செயல்படுவதைக் காணலாம்.

```
print (df.sum())


Tamil   72.000000
English 86.666667
Maths   67.333333
Science 76.333333
Social  72.666667
dtype: float64
```

□□□ □□□□□□□□□ □□□□□□□ □□□□□□□□□ □□□□□□□□□ □□□□□ replace() □□□□□□□□□□□□□.

```
print (df.replace({72:100}))


       Tamil  English   Maths     Science   Social
Ramesh NaN    NaN       NaN       NaN       NaN
Suresh NaN    NaN       NaN       NaN       NaN
Kamesh 100.0  86.666667 67.333333 76.333333 72.666667
```

11 [illegible] Pandas-11

11.1 Handling DateTime

[illegible]

```
import pandas as pd


df = pd.DataFrame({'Name':
['Mahesh','Elango','Kadiresan','Sundar','Kumar'],'Joining_date':
['2020.01.14', '2020, 12, 23', 'Nov 19, 2020', '2020, 04,
06','2020.06.30']})


print (df)


df['Joining_date'] = pd.to_datetime(df['Joining_date'])


print (df)


df['confirmation_date'] = df['Joining_date']+pd.Timedelta(days=180)


print (df)




print (pd.Timedelta(days=180))


print (pd.Timedelta('3 days 5 hours 45 minutes 50 seconds'))


print (pd.Timedelta(5,unit='h'))


l1 = [pd.Timedelta(days=i) for i in range(3)]


print (pd.Series(l1))
```

```
df['Allocation_date'] = pd.date_range('01/01/2021', periods=5)

print (df)



print (pd.date_range("11:00", "13:30", freq="30min"))

print (pd.date_range('27-Sep-2017', '17-Oct-2017',freq='3D'))



df['Allocation_date'] = pd.bdate_range('01/01/2021', '01/07/2021')

print (df)



print (pd.bdate_range('27-Sep-2017', '17-Oct-2017'))



print (pd.date_range('27-Sep-2017', '17-Dec-2017',freq='M'))

print (pd.period_range('27-Sep-2017', '17-Dec-2017',freq='M'))



df['Today_date'] = pd.datetime.now()

print (df)



df['Experience'] = df['Today_date'] - df['Joining_date']
```

```
print (df['Experience'])



print (pd.datetime.now())

print (pd.datetime.now().strftime("%Y-%m-%d"))

print (pd.Timestamp('2021-03-10'))

print (pd.Timestamp(1372898493,unit='s'))
```

Code link 11_pandas_datetime.py

11.1.1 Supported format

இங்கு 5 நபர்களின் பெயர் மற்றும் அவர்கள் வேலைக்குச் சேர்ந்த தேதி ஆகியவற்றைக் கொண்ட ஒரு டேட்டாஃப்பிரேம் உருவாக்கப்பட்டுள்ளது.

இத்தேதியிலிருந்து சரியாக 6 மாதங்கள் கழித்து அவர்களது வேலையை உறுதி செய்வதற்கான நாளைக் கணக்கிட வேண்டுமெனில், முதலில் அவர்கள் வேலைக்குச் சேர்ந்த தேதியின் அமைப்பினை pandas ஏற்றுக் கொள்ளக் கூடிய வடிவில் மாற்ற வேண்டும். இதற்கு உதவுவதே to_datetime ஆகும்.

இங்கு பல்வேறு வடிவில் கொடுக்கப்பட்ட தேதிகள் அனைத்தும் yyyy-mm-dd ஆகிய வடிவில் மாற்றப்பட்டுள்ளன.

```
df = pd.DataFrame({'Name':
['Mahesh','Elango','Kadiresan','Sundar','Kumar'],'Joining_date':
['2020.01.14', '2020, 12, 23', 'Nov 19, 2020', '2020, 04,
06′,'2020.06.30']})


print (df)


  Name      Joining_date
0 Mahesh    2020.01.14
1 Elango    2020, 12, 23
2 Kadiresan Nov 19, 2020
```

```
3 Sundar    2020, 04, 06
4 Kumar     2020.06.30


df['Joining_date'] = pd.to_datetime(df['Joining_date'])
print (df)


  Name      Joining_date
0 Mahesh    2020-01-14
1 Elango    2020-12-23
2 Kadiresan 2020-11-19
3 Sundar    2020-04-06
4 Kumar     2020-06-30
```

11.1.2 Timedelta

Timedelta என்பது நேரங்களில் உள்ள கால வேறுபாடுகளைக் காட்ட உதவும் அலகு முறை ஆகும். ஒவ்வொருவருடைய வேலை உறுதி நாளையும் கணக்கிட அவர்கள் வேலைக்குச் சேர்ந்த தேதியிலிருந்து 180 நாட்களைக் கூட்ட இந்த அலகு பயன்பட்டுள்ளது. அவ்வாறு கூட்டி கண்டுபிடித்த நாளானது confirmation date எனும் Column-ன் கீழ் சேமிக்கப்படுகிறது .

```
df['confirmation_date'] = df['Joining_date']+pd.Timedelta(days=180)
print (df)


  Name      Joining_date confirmation_date
0 Mahesh    2020-01-14   2020-07-12
1 Elango    2020-12-23   2021-06-21
2 Kadiresan 2020-11-19   2021-05-18
3 Sundar    2020-04-06   2020-10-03
4 Kumar     2020-06-30   2020-12-27
```

Time delta-வை நாட்கள், மணி, நேரம், நிமிடம் என பல்வேறு அளவுகளில் அமைக்கலாம். அதற்கான உதாரணங்கள் கீழே கொடுக்கப்பட்டுள்ளது .

```
print (pd.Timedelta(days=180))
180 days 00:00:00


print (pd.Timedelta('3 days 5 hours 45 minutes 50 seconds'))
3 days 05:45:50
```

```
print (pd.Timedelta(5,unit='h'))
0 days 05:00:00


l1 = [pd.Timedelta(days=i) for i in range(3)]
print (pd.Series(l1))


0 0 days
1 1 days
2 2 days
dtype: timedelta64[ns]
```

11.1.3 Date Range

கொடுக்கப்பட்ட எல்லைக்கு ஏற்றவாறு தேதிகளை உருவாக்க date_range எனும் function பயன்படும்.

இங்கு ஜனவரி 1-லிருந்து துவங்கி தொடர்ச்சியான 5 தேதியினை Allocation date எனும் column-ன் கீழ் அமைப்பதைக் காணலாம்.

பொதுவாக சனி, ஞாயிறு ஆகிய விடுமுறை நாட்களில் அல்லகேஷன் தேதி அமையாது. இதுபோன்ற இடங்களில் பயன்படுத்த உதவுவதே bdate_range() ஆகும். இது பிசினஸ் நாட்களை மட்டும் கணக்கில் கொண்டு தேதிகளை அமைக்கும்.

```
df['Allocationdate'] = pd.date_range('01/01/2021′, periods=5)
print (df)


  Name      Joining_date confirmation_date Allocation_date
0 Mahesh    2020-01-14   2020-07-12        2021-01-01
1 Elango    2020-12-23   2021-06-21        2021-01-02
2 Kadiresan 2020-11-19   2021-05-18        2021-01-03
3 Sundar    2020-04-06   2020-10-03        2021-01-04
4 Kumar     2020-06-30   2020-12-27        2021-01-05
```

இதன் frequency எனும் பண்பு எவ்வளவு நாட்கள் அல்லது நேரங்கள் இடைவெளி விட்டு அடுத்த தேதியினை உருவாக்க வேண்டும் என்பதைக் குறிக்கிறது. கீழே உள்ள எடுத்துக்காட்டில் அரை மணி நேரத்திற்கு ஒரு முறையும் மூன்று நாட்களுக்கு ஒரு முறையும் தேதிகள் உருவாக்கிக் காட்டப்பட்டுள்ளன.

```
print (pd.date_range(“11:00", “13:30″, freq="30min"))
```

```
DatetimeIndex(['2021-05-06 11:00:00', '2021-05-06 11:30:00','2021-05-06 12:00:00', '2021-05-06 12:30:00','2021-05-06 13:00:00', '2021-05-06 13:30:00'],dtype='datetime64[ns]', freq='30T')


print (pd.date_range('27-Sep-2017′, '17-Oct-2017′,freq='3D'))


DatetimeIndex(['2017-09-27', '2017-09-30', '2017-10-03', '2017-10-06','2017-10-09', '2017-10-12', '2017-10-15′],
dtype='datetime64[ns]', freq='3D')
```

11.1.4 Business dates

கொடுக்கப்பட்ட எல்லைக்கு ஏற்றவாறு சனி, ஞாயிறு ஆகிய விடுமுறை நாட்களை விடுத்து தேதிகளை உருவாக்க bdate_range எனும் function பயன்படும். இங்கு ஜனவரி 1-லிருந்து துவங்கி 7-ஆம் தேதி வரை சனி, ஞாயிறு ஆகிய தேதிகளைத் தவிர்த்து தொடர்ச்சியான 5 தேதிகள் Allocation date எனும் column-ன் கீழ் அமைவதைக் காணலாம்.

```
df['Allocationdate'] = pd.bdate_range('01/01/2021′, '01/07/2021′)
print (df)


  Name      Joining_date confirmation_date Allocation_date
0 Mahesh    2020-01-14   2020-07-12        2021-01-01
1 Elango    2020-12-23   2021-06-21        2021-01-04
2 Kadiresan 2020-11-19   2021-05-18        2021-01-05
3 Sundar    2020-04-06   2020-10-03        2021-01-06
4 Kumar     2020-06-30   2020-12-27        2021-01-07
```

இதற்கான மற்றுமொரு எடுத்துக்காட்டு பின்வருமாறு.

```
print (pd.bdate_range('27-Sep-2017′, '17-Oct-2017′))


DatetimeIndex(['2017-09-27', '2017-09-28', '2017-09-29', '2017-10-02','2017-10-03', '2017-10-04', '2017-10-05', '2017-10-06','2017-10-09', '2017-10-10', '2017-10-11', '2017-10-12′,'2017-10-13', '2017-10-16', '2017-10-17'],
dtype='datetime64[ns]', freq='B')
```

11.1.5 Period Range

period_range என்பது மாதங்களை அடிப்படையாகக் கொண்டு அமையும். date_range என்பதிலும் freq=M எனக் கொடுக்கலாம். ஆனால் date_range உள்

கொடுக்கப்படுகின்றன கடைசி எல்லை மாதத்தின் நடுவில் உள்ள ஏதாவது ஒரு தேதியைக் கொண்டிருந்தால் அக்கடைசி மாதத்தைக் கணக்கில் எடுத்துக் கொள்ளது. ஆனால் period_range என்பது மாதத்தின் எந்த ஒரு தேதியைக் கொண்டிருந்தாலும் அதனைக் கணக்கில் எடுத்துக் கொள்ளும்.

கீழே உள்ள எடுத்துக்காட்டில் , 17-Dec-2017 எனும் எல்லை date_range உள் கொடுக்கப்படும்போது திசம்பர் மாதம் விடுபடுவதைக் காணவும். ஆனால் அதே தேதி period_range உள் கொடுக்கப்படும்போது திசம்பர் மாதம் இடம்பெறுவதைக் காணவும்.

```
print (pd.date_range('27-Sep-2017′, '17-Dec-2017′,freq='M'))


DatetimeIndex(['2017-09-30', '2017-10-31', '2017-11-30′],
dtype='datetime64[ns]', freq='M')


print (pd.period_range('27-Sep-2017′, '17-Dec-2017′,freq='M'))


PeriodIndex(['2017-09', '2017-10', '2017-11', '2017-12′],
dtype='period[M]', freq='M')
```

11.1.6 DateTime & Timestamp

அடுத்ததாக நமது டேட்டாஃப்பிரேமில் Todaydate எனும் column-ஐ இணைத்து அதில் தற்போதைய தேதியை சேமிக்கின்றோம். ஏனெனில் இதனை அடிப்படையாகக் கொண்டு தான் ஒருவருடைய அனுபவத்தை கணக்கிடப் போகிறோம். அது பின்வருமாறு.

```
df['Todaydate'] = pd.datetime.now()
print (df)


  Name   Joining_date … Allocation_date Today_date
0 Mahesh 2020-01-14   … 2021-01-01      2021-05-06 14:21:59.847837
1 Elango 2020-12-23   … 2021-01-04      2021-05-06 14:21:59.847837
2 Kadir  2020-11-19   … 2021-01-05      2021-05-06 14:21:59.847837
3 Sundar 2020-04-06   … 2021-01-06      2021-05-06 14:21:59.847837
4 Kumar  2020-06-30   … 2021-01-07      2021-05-06 14:21:59.847837
[5 rows x 5 columns]


df['Experience'] = df['Todaydate'] – df['Joiningdate']
print (df['Experience'])


0 478 days 14:21:59.847837
1 134 days 14:21:59.847837
```

```
2 168 days 14:21:59.847837
3 395 days 14:21:59.847837
4 310 days 14:21:59.847837
Name: Experience, dtype: timedelta64[ns]
```

datetime வெளிப்படுத்துகின்ற தற்போதைய நேரத்தை நாம் விரும்பும் வடிவில் அமைக்க விரும்பினால் strftime() என்பது பயன்படும்.

Timestamp() என்பது கொடுக்கப்பட்டுள்ள தேதியை நேர மதிப்பீட்டுடன் இணைத்து வெளிப்படுத்த உதவும். அவ்வாறே ஒரு epoch time-ஐ மாற்றிக் கொடுக்கவும் உதவும்.

```
print (pd.datetime.now())
2021-05-06 14:21:59.862609


print (pd.datetime.now().strftime(“%Y-%m-%d"))
2021-05-06


print (pd.Timestamp('2021-03-10'))
2021-03-10 00:00:00


print (pd.Timestamp(1372898493,unit='s'))
2013-07-04 00:41:33
```

12 எளிய தமிழில் Pandas-12

12.1 Handling Categorical data

ஒருவருடைய பாலினம், ரத்தவகை என்பது போன்ற மதிப்புகளைக் குறிப்பிடும் போது ஒருசில குறிப்பிட்ட மதிப்புகளையே திரும்பத் திரும்ப அளிக்க வேண்டிவரும் . இதுபோன்ற சமயங்களில் string என்பதற்கு பதிலாக category எனும் தரவுவகையின் கீழ் அமைத்தால் நினைவகப் பகுதியை சற்று சேமிக்கலாம். எனவேதான் இந்த category-ஆனது hybrid வகை datatype என்று அழைக்கப்படுகிறது.

இத்தகைய categorical டேட்டாவை வைத்து எழுதப்பட்ட உதாரண நிரல் பின்வருமாறு.

```
import pandas as pd



d = {'Names' :
pd.Series(['Mahesh','Yazhini','Kadiresan','Malathi','Kumar','Sujith']),


'Gender' :
pd.Series(['Male','Trans','Male','Female','Male','Trans'],dtype="category")}


df = pd.DataFrame(d)



print (df['Names'])


print (df['Gender'])


print (df['Gender'].cat.remove_categories(['Trans'])) # add_categories()


print (df['Gender'].cat.categories)



c = pd.Categorical(['AB+', 'B+', 'AB+', 'O+', 'B-', 'AB-','B+'])


df['Blood_group'] = pd.Series(c)
```

```
print (df['Blood_group'])



c = pd.Categorical(['AB+', 'B+', 'AB+', 'O+', 'B-', 'AB-','B+'],['O+','B+'])

df['Blood_group'] = pd.Series(c)

print (df['Blood_group'])

print (c.ordered)

df['Mid Year points'] = pd.Series(["50","90","80","50","120","100"])

df['Year End points'] = pd.Series(["80","50","80","90","100","50"])



df['Mid Year points'] = df['Mid Year points'].astype("category",
categories=["50","80","90"], ordered=True)

df['Year End points'] = df['Year End points'].astype("category",
categories=["50","80","90"], ordered=True)

print (df['Mid Year points']>df['Year End points'])

print (df)
```

code link – 12_pandas_categoricaldata.py

[illegible] 6 [illegible] [illegible] [illegible] [illegible] [illegible] Names, Gender [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].
[illegible] Gender [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] dtypes=category [illegible] [illegible].

```
d = {'Names' :
pd.Series(['Mahesh','Yazhini','Kadiresan','Malathi','Kumar','Sujith']),
'Gender' :
pd.Series(['Male','Trans','Male','Female','Male','Trans'],dtype="category")}
df = pd.DataFrame(d)
```

ஆகவேதான் Names-ஐப் பிரிண்ட் செய்து பார்க்கும்போது அதனுடைய dtype மதிப்பு object என வெளிப்படுவதையும், Gender-ஐப் பிரிண்ட் செய்யும் போது அதனுடைய மதிப்பு category என வெளிப்படுவதையும் காணலாம்.

```
print (df['Names'])


0 Mahesh
1 Yazhini
2 Kadiresan
3 Malathi
4 Kumar
5 Sujith
Name: Names, dtype: object
```

மேலும் பல முறை திரும்பத் திரும்ப வந்துள்ள மதிப்புகளை ஒருமுறை மட்டும் எடுத்துக் காட்டி மொத்தம் இத்தனை மதிப்புகளே இதில் இடம் பெற்றுள்ளன என்பதையும் வெளிப்படுத்தும்.

இங்கு Male, Female, Trans ஆகிய மூன்று மதிப்புகளே திரும்பத் திரும்ப வந்துள்ளன என்பதை வெளிப்படுத்தியுள்ளது.

```
print (df['Gender'])


0 Male
1 Trans
2 Male
3 Female
4 Male
5 Trans
Name: Gender, dtype: category
Categories (3, object): [Female, Male, Trans]
```

மொத்தமுள்ள 3 மதிப்புகளில் ஏதேனும் ஒரு மதிப்பினை கேட்டகிரியில் இருந்து நீக்க விரும்பினால் அது பின்வருமாறு அமையும். இங்கு Trans எனும் மதிப்பு நீக்கப்பட்டுள்ளது . ஆகவே இந்த மதிப்பு அமைந்துள்ள இடங்களில் NaN என்பதை வெளிப்படுத்தும்.

```
print (df['Gender'].cat.remove_categories(['Trans'])) #
add_categories()


0 Male
1 NaN
2 Male
3 Female
4 Male
5 NaN
Name: Gender, dtype: category
Categories (2, object): [Female, Male]
```

ஒரு கேட்டகிரியில் மொத்தம் என்னென்ன மதிப்புகள் உள்ளன என்பதைக் காண்பதற்கான கட்டளை பின்வருமாறு.

```
print (df['Gender'].cat.categories)
Index(['Female', 'Male', 'Trans'], dtype='object')
```

ஒரு series-க்குள் மதிப்புகளைக் கொடுத்து அதன் dtype=category என வரையறுப்பதன் மூலம் எவ்வாறு categorical டேட்டாவை உருவாக்கலாம் என்று பார்த்தோம். இவ்வாறு அல்லாமல் நேரடியாக pd.categorical என்பதற்குள் மதிப்புகளைக் கொடுத்து அதனை series-க்குள் செலுத்துவதன் மூலமும் இத்தகைய டேட்டாவை உருவாக்கலாம்.

இங்கு Blood_group எனும் டேட்டா இம்முறையில் உருவாக்கப்பட்டுள்ளது.

```
c = pd.Categorical(['AB+', 'B+', 'AB+', 'O+', 'B-', 'AB-','B+'])


df['Blood_group'] = pd.Series(c)
print (df['Blood_group'])


0 AB+
1 B+
2 AB+
3 O+
4 B-
5 AB-
Name: Blood_group, dtype: category
Categories (5, object): [AB+, AB-, B+, B-, O+]
```

பொதுவாக pd.Categorical என்பதற்குள் கொடுக்கப்பட்ட லிஸ்டில் உள்ள unique மதிப்புகளை எடுத்து கேட்டகிரியில் உள்ள மதிப்புகள் அமையும். இவ்வாறு அல்லாமல் ஒரு குறிப்பிட்ட மதிப்புகளை மட்டும் கொடுத்து அதனை கேட்டகிரியின் மதிப்பாக அமைக்க விரும்பினால் அது பின்வருமாறு.

இங்கு pd.Categorical என்பதற்குள் உள்ள முதல் லிஸ்டில் பல்வேறு மதிப்புகள் கொடுக்கப்பட்டாலும், அதனைத் தொடர்ந்து உள்ள அடுத்த லிஸ்டில் ['O+','B+'] மதிப்புகளை மட்டும் கேட்டகிரியின் மதிப்பாக அமைக்குமாறு கொடுத்துள்ளோம்.

எனவேதான் இம்மதிப்புகள் தவிர மற்ற மதிப்புகளுக்கு NaN வெளிப்படுத்தியுள்ளதைக் காணலாம்.

```
c = pd.Categorical(['AB+', 'B+', 'AB+', 'O+', 'B-', 'AB-','B+'],
['O+','B+'])
df['Blood_group'] = pd.Series(c)
print (df['Blood_group'])


0 NaN
1 B+
2 NaN
3 O+
4 NaN
5 NaN
Name: Blood_group, dtype: category
Categories (2, object): [O+, B+]
```

Ordered எனும் பண்பினைப் பயன்படுத்தி கேட்டகிரி உருவாக்கப்பட்டுள்ளதா இல்லையா என்பதைக் கண்டுபிடிக்க பின்வருமாறு கொடுக்கலாம்.

```
print (c.ordered)
False
```

இப்பண்பினைப் பயன்படுத்தி அடுத்து இரு series-ஐ உருவாக்கி பார்க்கலாம்.

ஒரு series-ஐ உருவாக்கும்போது dtype=category எனக் கொடுக்கலாம் அல்லது pd.Categorical எனக் கொடுத்து கேட்டகிரியை உருவாக்கிய பின்னர் series-ஐ

உருவாக்கலாம். இவ்விரண்டு முறையிலும் அல்லாமல் astype எனக் கொடுத்து ஒரு series-ஐ உருவாக்கிய பின்னரும் கேட்டகிரியாக மாற்றலாம்.

இங்கு ஒருவர் ஆண்டின் இடையில் எவ்வளவு மதிப்புகள் பெற்றுள்ளார் மற்றும் ஆண்டின் இறுதியில் எவ்வளவு மதிப்புகள் பெற்றுள்ளார் என்பதை வைத்து இரண்டு சீரீஸ் உருவாக்கப்பட்டுள்ளது .

```
df['Mid Year points'] = pd.Series([“50″,"90″,"80″,"50″,"120″,"100"])
df['Year End points'] = pd.Series([“80″,"50″,"80″,"90″,"100″,"50"])
```

பின்னர் astype மூலம் கேட்டகிரி உருவாக்கப்பட்டு அதன் மதிப்புகளாக [“50″,"80″,"90"] ஆகியவற்றை மட்டும் எடுத்துக் கொள்ளுமாறு கூறப்பட்டுள்ளது. இதனை உருவாக்கும் போது ordered=True எனக் கொடுத்து உருவாக்கியுள்ளோம். இதன் மூலம் இவ்விரண்டு series-ஐயும் ஒப்பிடும் வாய்ப்பு நமக்குக் கிடைக்கிறது.

```
df['Mid Year points'] = df['Mid Year points'].astype(“category",
categories=[“50″,"80″,"90"], ordered=True)
df['Year End points'] = df['Year End points'].astype(“category",
categories=[“50″,"80″,"90"], ordered=True)
```

இங்கு ஒருவர் ஆண்டின் இடையில் பெற்ற மதிப்புகளைவிட ஆண்டின் இறுதியில் அதிக மதிப்பெண்கள் பெற்றுள்ளாரா இல்லையா என்பது கண்டுபிடிக்கப்பட்டுள்ளது .

```
print (df['Mid Year points']>df['Year End points'])


0 False
1 True
2 False
```

```
3 False
4 False
5 False
Name: Year End points, dtype: bool
```

13 [illegible] Pandas-13_Final

13.1 Real-time Example

[illegible] id, [illegible], [illegible], [illegible], [illegible]. [illegible].

	category	rating	telecasted_date	program_id	score
0	news	3	12/26/2020 12:06	67547	65
1	news	4	10/15/2020 12:06	87465	75
2	news	3	12/26/2020 12:06	88756	87
3	news	3	12/26/2020 12:06	98456	56
4	news	4	10/15/2020 12:06	90908	46
5	news	3	12/26/2020 12:06	34355	98
6	news	3	12/26/2020 12:06	87643	95
7	news	3	12/26/2020 12:06	68864	100
8	Variety shows	4	12/26/2019 12:06	23254	78
9	Variety shows	4	12/26/2019 12:06	88997	56
10	Variety shows	2	8/12/2019 12:06	57636	87
11	Variety shows	5	6/17/2019 12:06	56643	56
12	Variety shows	4	12/26/2019 12:06	67765	45
13	Variety shows	4	12/26/2019 12:06	65499	64
14	Variety shows	3	6/19/2019 12:06	76439	86
15	Variety shows	4	12/26/2019 12:06	77865	57
16	Variety shows	5	6/17/2019 12:06	76638	87
17	Variety shows	4	12/26/2019 12:06	89097	75
18	Variety shows	3	6/19/2019 12:06	87539	89
19	Variety shows	4	12/26/2019 12:06	78754	100

[illegible] raw [illegible] logic-[illegible].

இங்கும் பின்வரும் விதிகளின் அடிப்படையில் புதிய தரவுகளை உருவாக்கப் போகிறோம்.

1. category, rating ஆகியவற்றில் உள்ள தரவுகளை _ மூலம் இணைத்து code எனும் புதிய column – இன் கீழ் வெளிப்படுத்த வேண்டும். எடுத்துக்காட்டாக category=news, rating=3 எனில் code=news_3 என இருக்க வேண்டும்.

2. telecasted_date என்ற column-ல் உள்ள தேதியின் அடிப்படையில் telecasted_quarter என்ற column-ஐ உருவாக்க வேண்டும். எடுத்துக்காட்டாக 12/26/2020 எனும் தேதியை வைத்து உருவாக்கப்படும் quarter-ன் மதிப்பு '2020 Quarter 4' எனும் வடிவில் இருக்க வேண்டும்.

3. ஒரு category-யின் கீழ் ஒரே ரேட்டிங்கை பெற்றிருக்கும் புரோகிராம்களின் எண்ணிக்கை 5-க்கு கீழ் இருந்தால் avg_score எனும் தலைப்பின் கீழ் அவை அனைத்தும் பெற்றுள்ள ஸ்கோரின் சராசரியைக் கண்டுபிடித்து வெளிப்படுத்த வேண்டும். 5-க்கும் மேல் இருந்தால் முதல் ஐந்து புரோகிராம்கள் பெற்றுள்ள ஸ்கோரின் சராசரியை மட்டும் கண்டுபிடித்து வெளிப்படுத்த வேண்டும் . எடுத்துக்காட்டாக இங்கு நியூஸ் எனும் கேட்டகிரியின் கீழ் 3 எனும் ரேட்டிங்கை பெற்றிருக்கும் புரோகிராம்களின் எண்ணிக்கை மொத்தம் 6. ஆகவே முதல் 5 புரோகிராம்களின் சராசரியான (65+87+56+98+95)/5 = 80.2 ஆகியவற்றை வெளிப்படுத்த வேண்டும். அதுவே நியூஸ் எனும் கேட்டகிரியின் கீழ் 4 எனும் ரேட்டிங்கை பெற்றிருக்கும் புரோகிராம்களின் எண்ணிக்கை மொத்தம் 2 தான். ஆகவே அந்த இரண்டின் சராசரியை மட்டும் வெளிப்படுத்தினால் போதும்.

மேற்குறிப்பிட்ட மூன்று லாஜிக் ஐயும் பாண்டாஸ் மூலம் செய்து வெளிப்படுத்துவதற்கான நிரல் பின்வருமாறு அமையும்.

```
import pandas as pd

from datetime import datetime,timedelta

import numpy as np



df = pd.read_csv('./13_input_data.csv')
```

```
print (df)



pd.set_option("display.max_columns",8)



df1 = pd.DataFrame()

df1['code'] = df['category'].astype(str)+'_'+df['rating'].astype(str)

df1['rating'] = df['rating']

df1['year'] = pd.DatetimeIndex(df['telecasted_date']).year

df1['telecasted_quarter'] = df1['year'].astype(str)+' Quarter
'+pd.DatetimeIndex(df['telecasted_date']).quarter.astype(str)

df1['score'] = df['score']

print (df1.head(3))



df2 = df1.groupby(['code','rating','year','telecasted_quarter'])
['rating'].count().reset_index(name="total_programs")

df2['greater_than_5'] = df2['total_programs'].apply(lambda x: 'YES'
if x >= 5 else 'No')

df3 = df1.groupby(['code','rating','year','telecasted_quarter'])
['score'].mean().reset_index(name="all")

df2['all'] = df3['all']
```

```
df4 = df1.groupby(['code','rating','year','telecasted_quarter'])
['score']

l1 = []

for i,j in df4:
    y = float(j[0:5].mean())
    l1.append(y)

df2['first_5'] = pd.Series(l1)

print (df2)

l2 = []

for i,j,k in zip(df2['all'],df2['first_5'],df2['greater_than_5']):
    if k=='YES':
         l2.append(j)
    else:
         l2.append(i)

df2['avg_score'] = pd.Series(l2)

df2 = df2[['code','telecasted_quarter','avg_score']]



print (df2)
```

Code link 13_pandas_realtime_example.py

[illegible]:

1. [illegible] df [illegible].

பின் df1 எனும் ஒரு empty டேட்டாஃப்பிரேமை உருவாக்கி அதற்குள் code எனும் column-ன் கீழ் df-லிருந்து கேட்டகிரி மற்றும் ரேட்டிங் ஆகியவற்றில் உள்ள மதிப்புகளை எடுத்து அதனை underscore மூலம் இணைத்து சேமிக்கிறோம்.
பின் telecasted_date என்ற column-ல் இருக்கும் தேதியிலிருந்து year மற்றும் quarter ஆகியவற்றைப் பிரித்து அதனை நாம் விரும்பும் வடிவில் Quarter எனும் வார்த்தையையும் இணைத்து telecasted_quarter என்ற column-ன் கீழ் சேமித்துள்ளோம்.

பின் ஒரே வகையான கேட்டகிரி மற்றும் ரேட்டிங்கை குரூப் செய்து ஸ்கோரைக் கணக்கிட வேண்டும் என்பதால், இதற்குத் தேவையான rating, score ஆகியவற்றையும் df1-க்குள் சேமித்து உள்ளோம்.

```
df1 = pd.DataFrame()
df1['code'] = df['category'].astype(str)+'_'+df['rating'].astype(str)
df1['rating'] = df['rating']
df1['year'] = pd.DatetimeIndex(df['telecasted_date']).year
df1['telecasted_quarter'] = df1['year'].astype(str)+' Quarter
'+pd.DatetimeIndex(df['telecasted_date']).quarter.astype(str)
df1['score'] = df['score']
print (df1.head(3))


     code    rating  year   telecasted_quarter  score
0  news_3       3  2020     2020 Quarter 4     65
1  news_4       4  2020     2020 Quarter 4     75
2  news_3       3  2020     2020 Quarter 4     87
```

2. பின் ஒரே வகையான கேட்டகிரி மற்றும் ரேட்டிங்கை குரூப் செய்து, ஒவ்வொரு தனித்தனி குரூப்பிலும் எத்தனை புரோகிராம் உள்ளது எனக் கணக்கிட்டு அதனை total_programs எனும் தலைப்பின் கீழ் அமைக்கிறது. இந்த புதிய டேட்டாஃப்பிரேமின் பெயர் df2 ஆகும். பின் இந்த count 5- இக்கு மேல் இருந்தால் YES எனும் மதிப்பையும், கீழ் இருந்தால் NO எனும் மதிப்பையும் கொள்ளுமாறு 'greater_than_5' எனும் column உருவாக்கப்பட்டுள்ளது. பின் all எனும் column-ன் கீழ் கேட்டகிரி , ரேட்டிங் முறையில் குரூப் செய்யப்பட்ட அனைத்து score-ன் சராசரி மதிப்பும் கண்டுபிடிக்கப் பட்டுள்ளது . அதேபோல first_5 எனும் column-ன் கீழ் கேட்டகிரி , ரேட்டிங் முறையில் குரூப் செய்யப்பட்ட முதல் 5 ஸ்கோரின் சராசரி மதிப்பு கண்டுபிடிக்கப்பட்டுள்ளது.

```
df2 = df1.groupby(['code','rating','year','telecasted_quarter'])
['rating'].count().reset_index(name="total_programs")


df2['greater_than_5'] = df2['total_programs'].apply(lambda x: 'YES' if x >=
5 else 'No')
```

```
df3 = df1.groupby(['code','rating','year','telecasted_quarter'])
['score'].mean().reset_index(name="all")


df2['all'] = df3['all']


df4 = df1.groupby(['code','rating','year','telecasted_quarter'])['score']


l1 = []
for i,j in df4:
     y = float(j[0:5].mean())
     l1.append(y)


df2['first_5'] = pd.Series(l1)


print (df2)
```

```
             code  rating  year  telecasted_quarter  total_programs  \
0 Variety shows_2       2  2019     2019 Quarter 3               1
1 Variety shows_3       3  2019     2019 Quarter 2               2
2 Variety shows_4       4  2019     2019 Quarter 4               7
3 Variety shows_5       5  2019     2019 Quarter 2               2
4          news_3       3  2020     2020 Quarter 4               6
5          news_4       4  2020     2020 Quarter 4               2

  greater_than_5        all  first_5
0             No  87.000000     87.0
1             No  87.500000     87.5
2            YES  67.857143     60.0
3             No  71.500000     71.5
4            YES  83.500000     80.2
5             No  60.500000     60.5
```

3. [illegible] greater_than_5 [illegible] column-[illegible] [illegible] [illegible] YES [illegible], first_5 [illegible] column-[illegible] [illegible] [illegible] [illegible] [illegible], NO [illegible] all [illegible] column-[illegible] [illegible] [illegible] [illegible] [illegible], [illegible] [illegible] [illegible] avg_score [illegible] column-[illegible] [illegible]. [illegible] [illegible] [illegible] column-[illegible] [illegible] [illegible] [illegible].

```
l2 = []
for i,j,k in zip(df2['all'],df2['first_5'],df2['greater_than_5']):
    if k=='YES':
        l2.append(j)
    else:
        l2.append(i)
df2['avg_score'] = pd.Series(l2)
df2 = df2[['code','telecasted_quarter','avg_score']]
print (df2)
```

	code	telecasted_quarter	avg_score
0	Variety shows_2	2019 Quarter 3	87.0
1	Variety shows_3	2019 Quarter 2	87.5
2	Variety shows_4	2019 Quarter 4	60.0
3	Variety shows_5	2019 Quarter 2	71.5
4	news_3	2020 Quarter 4	80.2
5	news_4	2020 Quarter 4	60.5

எளிய தமிழில் Pandas Kaniyam.com

14முடிவுரை

இந்த நூல் Pandas க்கான ஒரு அறிமுகம் மட்டுமே. இவை பற்றி இணையத்தில் பல்வேறு பாடங்களும் காணொளிகளும் கிடைக்கின்றன. அவற்றை தொடர்ந்து படித்து, கற்று, பல்வேறு சாதனைகள் புரிய அனைவருக்கும் வாழ்த்துகள்.

15 □□□□□□□□□□□□□□ □□□ □□□□□□□□□□□□

□□. □□□□□□ □□□□□□□□□□ □□□□□□□ □□□□□□□□□□□□□□□ □□□□□ □□□□□□□□□ □□□□□□□.

https://freetamilebooks.com/authors/nithyaduraisamy/

எளிய தமிழில் Pandas Kaniyam.com

16கணியம் அறக்கட்டளை

16.1 தொலை நோக்கு - Vision

தமிழ் மொழி மற்றும் இனக்குழுக்கள் சார்ந்த மெய்நிகர்வளங்கள், கருவிகள் மற்றும் அறிவுத்தொகுதிகள், அனைவருக்கும் கட்டற்ற அணுக்கத்தில் கிடைக்கும் சூழல்

16.2 பணி இலக்கு - Mission

அறிவியல் மற்றும் சமூகப் பொருளாதார வளர்ச்சிக்கு ஒப்ப, தமிழ் மொழியின் பயன்பாடு வளர்வதை உறுதிப்படுத்துவதும், அனைத்து அறிவுத் தொகுதிகளும், வளங்களும் கட்டற்ற அணுக்கத்தில் அனைவருக்கும் கிடைக்கச்செய்தலும்.

மேற்கண்ட தொலை நோக்கு, பணி இலக்குகளை நோக்கி பயணிக்க, கணியம் குழுவினர், கணியம் அறக்கட்டளையைத் தொடங்கியுள்ளோம். கணியம் இதழில் கட்டுரைகள் வெளியிடுதல், FreeTamilEbooks.com தளத்தில் மின்னூல்கள் வெளியிடுதல், தமிழுக்கான கட்டற்ற மென்பொருட்கள், பிற வளங்கள் உருவாக்குதல், இவற்றுக்கான இணைய வளங்களைப் பேணுதல், விக்கி மூலம் தளத்தில் மின்னூல்களை சீராக்கி வெளியிடுதல், கட்டற்ற மென்பொருட்களுக்கு பரப்புரை நிகழ்ச்சிகள் நடத்துதல் ஆகிய பணிகளை செவ்வனே செய்ய கணியம் அறக்கட்டளை ஆவன செய்யும்.

மேற்கண்ட பணிகளை ஆதரிக்க உங்களை நன்கொடைகளை வேண்டுகிறோம். பின்வரும் வங்கிக் கணக்கில் உங்கள் நன்கொடைகளை செலுத்தி, KaniyamFoundation@gmail.com க்கு ஒரு மின்னஞ்சல் அனுப்ப வேண்டுகிறோம்.

மிக்க நன்றி!

Kaniyam Foundation
Account Number : 606101010050279
Union Bank Of India
West Tambaram, Chennai
IFSC - UBIN0560618